Calhoun
Institutional Archive of the Naval Postgraduate School

Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations 1. Thesis and Dissertation Collection, all items

1964

An experimental study of the uses of ternary logic in digital computers.

Friichtenicht, Richard D.

http://hdl.handle.net/10945/12178

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun

DUDLEY
KNOX
LIBRARY

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NPS ARCHIVE
1964
FRIICHTENICHT, R.

AN EXPERIMENTAL STUDY OF THE USES OF TERNARY LOGIC IN DIGITAL COMPUTERS

RICHARD D. FRIICHTENICHT

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

Library
U. S. Naval Postgraduate School
Monterey, California

25

AN EXPERIMENTAL STUDY OF THE USES OF TERNARY LOGIC IN DIGITAL COMPUTERS

Richard D. Friichtenicht

AN EXPERIMENTAL STUDY OF THE USES OF TERNARY LOGIC IN DIGITAL COMPUTERS

by

Richard D. Friichtenicht
Lieutenant, United States Navy

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

IN

ENGINEERING ELECTRONICS

United States Naval Postgraduate School
Monterey, California

1 9 6 4

NPS Archive
1964
Fruchtenicht, R.

[illegible]

Library
U. S. Naval Postgraduate School
Monterey, California

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

AN EXPERIMENTAL STUDY OF THE USES OF TERNARY LOGIC IN DIGITAL COMPUTERS

by

Richard D. Friichtenicht

This work is accepted as fulfilling
the thesis requirements for the degree of

MASTER OF SCIENCE
IN
ENGINEERING ELECTRONICS

from the
United States Naval Postgraduate School

ABSTRACT

Digital computers presently in production are all binary logic machines, in that they are built with elements that have two stable states. Greater efficiency in computer speed and hardware would be obtained if elements with other than two states were used. Attempts are now in progress to find devices that have this property. The next logical step beyond binary would be ternary. A study of ternary algebras is made with emphasis on computer applications. Functional completeness and expansion theorems are introduced to show their usefulness in computer design. An adder circuit using three level logic is described and a measure of effectiveness using cost and complexity as criteria is made. It can be predicated that, as the binary computer approaches its ultimate in speed, more attention will be placed on N-valued logic machines.

ACKNOWLEDGMENTS

The author wishes to express his appreciation to Dr. R. I. Tanaka and Yates A. Keir of the Electronics Sciences Laboratory of Lockheed Missiles & Space Company for their guidance in choosing a field of research and for supplying the author with references which were indispensable in his research.

TABLE OF CONTENTS

Section	Title	Page
1.	Forward	1
2.	Terminology	2
3.	Introduction	5
4.	Algebraic Properties	8
5.	Functional Completeness	14
6.	An Experimental Ternary Adder	19
7.	Expansion Theorems	26
8.	Experimental Results	32
9.	Conclusion	37
	Bibliography	49
Appendix I	Derivation of Two-valued Two-place Functions	51
Appendix II	Derivation of Three-valued One-place Functions	64

LIST OF TABLES

Table | Page
I. Functional Completeness of AND/OR Compositions 46
II. Functional Completeness of NOR Compositions 47
III. Functional Completeness of AND/OR/NOT Compositions 48

LIST OF ILLUSTRATIONS

Figure	Page
1. Binary and Ternary Truth Tables of Four Inputs	42
2. Venn Diagrams for Binary and Ternary	43
3. Logic Diagram for Ternary Full Adder	44
4. Circuit Diagram for Ternary Half Adder	45
Appendix I, Fig. 1	
Flow Chart for Deriving Two-valued Two-place Functions	54
Appendix II, Fig. 1	
Flow Chart for Deriving Three-valued One-place Functions	66

1. Forward

Binary digital computers have been in use now for several decades, and as the evolution from vacuum tubes to semiconductors for circuitry took place, extremely fast switching times were realized. As more uses are found for computers, ever faster computers are desired. Switching times seem to be approaching a plateau which is in the order of nanoseconds and it appears that a major scientific breakthrough will be necessary before a further reduction will take place. With this in mind, most research at the present time is centered around getting more efficient use of the computer by improving the input/output interface. As a limit is approached in this field, new methods will be sought. It seems logical that the next step would be to go to a system other than the relatively inefficient binary number system. Using one particular criteria of goodness, it can be shown that the base three number system may be the most efficient since it is the closest whole number to the number "e" [7]. One might argue that the base ten system should be used since it is the system that is in most common usage. In any case, digital computers that operate on a basis other than the off-on or high-low system of binary computers do not appear to be too far from realization.

Present work in the field of ternary switching devices has not been caused by a necessity in the computer field, but rather by the desire to find uses for elements that have three stable states as their characteristic. Research is also attempting to find more elements that have three stable states. Once a device having desireable characteristics has been dis-

covered or invented, further research is necessary to see how it can be combined with devices having different characteristics.

2. Terminology

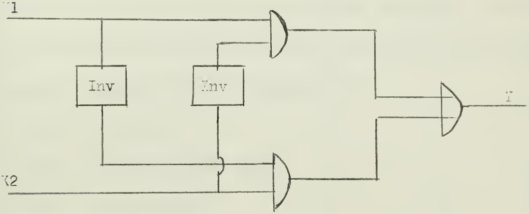
Ternary logic is logic that has three possible levels. These levels can be denoted by various methods such as high, medium, low or positive, zero, negative. Since a device that will do modular three addition will be discussed, the base three system of "0, 1, 2" will be used, with the zero to indicate low and the two to indicate high. A ternary device is an operator that transforms a number of inputs, each input having the value, "0, 1, or 2" into an output having a value "0, 1, or 2". A device that has three levels is referred to as a three-valued device and a device that has two inputs is referred to as a two-place device. A truth table relates any combination of input values to the corresponding output values of a device. Only two-place, three-valued devices will be discussed since any N-place device can be sub-divided into a combination of two-place devices by an iterative process.

Two-valued Boolean algebra will be used extensively for examples, references, and comparison purposes. An example of a two-place, two-valued device would be the OR operator. Capital letters will be used for operators as they are defined to distinguish them from common grammatical usage. The truth table for the OR device, using one for the high value, is given below:

		X1	
		0	1
X2	0	0	1
	1	1	1

The truth table can be simplified to just show the possible outputs. This is called a composition and in this case is denoted as (0111).

There are many methods of illustrating a logic circuit. Using standard symbology, the circuit below can be completely defined by any of the following equations or expressions:



(1). $Y = X1\overline{X2} + \overline{X1}X2$

(2). Y = (X1 AND NOT X2) OR (NOT X1 AND X2)

(3). Y =

	0	1
0	0	1
1	0	0

OR

	0	1
0	0	0
1	1	0

(4). Y = (0100) OR (0010)

(5). Y = (0100)

	0	1
0	0	1
1	1	1

(0010)

(6). Y = (0110)

(7). Y =

	0	1
0	0	1
1	1	0

For binary work, equation (1) has been by far the most popular form. But, this method means that all operations must be described in terms of the basic three Boolean operations of AND, OR and NOT. There are several reasons for this, the primary being that most computers are built with diodes and transistors that have these compositions as their characteristics. If a device were available that had two inputs, X1 and X2, and one output, Y, and if this device did all of the three operations in one easy step, the algebraic expression could be simplified to:

(8). $Y = X1 \oplus X2$

(9). Y = X1 EXCLUSIVE OR X2

(10). Y = X1 (0110) X2

(11). $$Y = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array}$$

which is basically the same as equations (6) and (7). If one were not familiar with the standard notation of ⊕ and "EXCLUSIVE OR", then the next simplest form, other than reverting to the Boolean functions, would be equations (10) and (11). In ternary logic, many compositions are not named and also the Boolean operations of AND, OR and NOT lose their meaning, therefore, algebraic expressions are often placed in the form of equations (4), (5), and (6).

It is important to stress the point that the terms: composition, operator, device, and function may all be used interchangeably without loss of meaning.

3. Introduction

In order to become familiar with ternary logic, one must have some idea of what is to be gained by using it. The fastest approach appears to be that of comparing numbers. Below are tabulated some representative numbers written in base ten, base two, and base three:

base 10	base 2	base 3	base 10	base 2	base 3
0	000	000	15	01111	0120
1	001	001	20	10100	0202
2	010	002	25	11001	0221
3	011	010	30	11110	1010
4	100	011	35	100011	1022
5	101	012	40	101000	1111
6	110	020	45	101101	1200
7	111	021	50	110010	1212
8	1000	022	60	111100	2020
9	1001	100	70	1000110	2121
10	1010	101	80	1010000	2222
11	1011	102	90	1011010	10100
12	1100	110	100	1100100	10201
13	1101	111			

Looking at the number 100, it is noted that it takes three digits to represent it in base ten, five digits in base three, and seven digits in base two. From this, two things are pointed out; there is a sizable decrease in digits when going from base two to base three; but, it is

necessary to increase the base number quite a bit before getting any corresponding improvement. There is actually a 59% improvement in base three versus base two.

Unfortunately, the complexity of the algebra for base three systems does not increase by a mere 59%; but rather, it increases many-fold. Truth tables for the binary and ternary situations will point out the increased complexity.

	0	1
0	a	b
1	c	d

	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i

In the binary case, "a" can take on two values, zero or one, as can "b", "c", and "d". Therefore, there are 2x2x2x2 = 16 possible compositions or operations. In the ternary case, "a" can take on three values, zero, one or two, as can "b", "c", "d", etc. There are now 3x3x3x3x3x 3x3x3x3 = 19,683 possible compositions. This points out the first major drawback in using ternary systems.

When confronted with such a large number of compositions, each representing a possible state or output, it is realized that some method must be employed to put them in a workable order. There are two primary methods used to approach this problem. One is to categorize the compositions based on algebraic properties and the second is to test groups of compositions for functional completeness. They will be discussed in the following sections.

Before going into specific detail on the two-place three-valued

functions, several interesting and useful features of ternary algebra will be pointed out.

There are several methods of representing the NOT composition in binary. One method shown below is the normal truth table form. Another method is a modified or shortened truth table also shown below. This device operates on one input only giving one output. This is commonly called a permutation of the input. There are 27 possible permutations of one input in ternary. One example ternary permutation is shown below in normal form and in shortened form.

	0	1
0	1	0
1	1	0

in	out
0	1
1	0

	0	1	2
0	1	2	0
1	1	2	0
2	1	2	0

in	out
0	1
1	2
2	0

Just as truth tables are extendable to any number of inputs in binary, so are truth tables extendable to any number of inputs in ternary. Fig. (1) shows two truth tables; the first truth table is for four inputs in binary and the second for four inputs in ternary. The ternary truth table could be extended to more inputs with the size of the table increasing by a factor of three for each additional input.

Venn diagrams are also extendable to three level logic. In order to have three levels, two circles are required; inside the inner circle is

considered a two, inside the outer circle but not the inner circle is considered a one, and outside the outer circle is considered a zero. Fig. (2) shows several examples of the binary Venn diagram and also shows several examples of the ternary Venn diagram.

4. Algebraic Properties

The first method that attempts to put the compositions in some sort of order is by grouping various compositions into classes according to properties of: commutativity, associativity, idempotency, truth values, and isomorphism. Once this is done, then one can discuss a class of compositions which may contain upwards of six compositions, rather than each composition separately. A good deal of work has been done with ternary compositions in this regard. Ref. [9] lists all commutative compositions by class and no attempt is made here to duplicate this work. However, the procedure used there will be used here on binary compositions. By comparing the algebraic properties of the compositions and functional completeness which is discussed in the next section, several conclusions can be made that will hold for ternary also.

Commutativity

From the completely general truth table shown below, a composition would be denoted as (p, q, r, s). Each of these letters would take on the value of "0" or "1" thus giving a total of 16 possible compositions. These compositions are listed below the truth table with their most common name.

	0	1
0	p	q
1	r	s

(0000)	ZERO	(0001)	AND
(0010)		(0011)	X2
(0100)		(0101)	X1
(0110)	EXCLUSIVE OR	(0111)	OR
(1000)	NOR	(1001)	COINCIDENCE
(1010)	NOT X1	(1011)	X2 IMPLIES X1
(1100)	NOT X2	(1101)	X1 IMPLIES X2
(1110)	NAND	(1111)	ONE

The commutative property is defined as X1oX2 = X2oX1, where "o" represents some arbitrary operation. Using standard notation of $X = 1$ and $\overline{X} = 0$, and using the OR operation as an example, the commutative property would state that;

$$0 + 1 = 1 + 0$$

Looking at the truth table, the commutative property states that it should make no difference by interchanging inputs to the device. Thus it should be possible to redraw the truth table except with the inputs, X1 and X2, reversed and the truth table should not be changed. The (0100) composition is one that does not fit this criterion, for it states that if $X1 = 1$ and $X2 = 0$ a "1" will result, but if $X2 = 1$ and $X1 = 0$ a "0" will result. A constraint has therefore been placed on the general truth table. For a composition to be commutative, "q" must equal "r". By making this constraint, only eight compositions are commutative:

(0000) (0001) (0110) (0111)

(1000) (1001) (1110) (1111)

Associativity

The associative property is defined as $(X1 \circ X2) \circ X3 = X1 \circ (X2 \circ X3)$. To check this property, two devices with the same composition would be needed; the first operating on X1 and X2 and the second operating on X3 and the output of the first. To be associative, the same result should be accomplished no matter how the inputs are interchanged. An example of this would be the OR operation where $(1 + 0) + 0 = 1 + (0 + 0) = 0 + (0 + 1)$. Looking at the modified general truth table below, we can state the constraints that have to be met for associativity:

	0	1
0	p	q
1	q	r

(1) $(1 + 0) + 0 = 1 + (0 + 0)$

$(\ q\ ) + 0 = 1 + (\ p\ )$

$q + 0 = 1 + p$

(2) $(1 + 1) + 0 = 1 + (1 + 0)$

$(\ r\ ) + 0 = 1 + (\ q\ )$

$r + 0 = 1 + q$

Several other constraints could be listed, but since the commutative property is now assumed, any other constraints would be redundant. For the present discussion, strictly the union (OR) is being used, but the same constraints hold true for intersections (AND) as well.

By assigning values to p, q, and r and checking the constraints, the compositions can be tested for associativity.

Case 1: (0001) For this composition, $p = 0$, $q = 0$, and $r = 1$.

Testing the left side of the first constraint:

$q + 0 = 0 + 0 = p = 0$

The right side of the constraint must give the same result:

$1 + p = 1 + 0 = q = 0$

So the first constraint is satisfied. Testing both sides of the second constraint:

left: $1 + q = 1 + 0 = q = 0$

right: $r + 0 = 1 + 0 = q = 0$

Both constraints are satisfied therefore the composition (0001) is associative.

Case2: (1000) For this composition, $p = 1$, $q = 0$, and $r = 0$.

A test is made on the first constraint:

left: $q + 0 = 0 + 0 = p = 1$

right: $1 + p = 1 + 0 = q = 0$

The two sides of the constraint do not give equal results, therefore the composition (1000) is not associative.

Checking all commutative compositions, there are only two that are also associative:

(0001) AND

(0111) OR

Idempotency

By definition, idempotency is where X1oX1 = X1. Using unions, the constraints can be established:

$0 + 0 = 0$

$1 + 1 = 1$

The letter q can take on either value and not affect the test, therefore,

there are two satisfactory compositions:

(0001) AND

(0111) OR

Unit truth value

By definition, a unit truth value, (e), is when X1o(e) = X1 for all values of X1. For a unit truth value of "0", r can take on values of either "0" or "1":

X1 = 0: $0 + 0 = p = 0$

X1 = 1: $1 + 0 = q = 1$

Thus, two compositions have unit truth values of "0", namely (0110) and (0111). For a unit truth value of "1", the following constraints are set up:

X1 = 0: $0 + 1 = q = 0$

X1 = 1: $1 + 1 = r = 1$

This leaves p to take on values of either "0" or "1". The two compositions that have unit truth values of "1" are (0001) and (1001). The four compositions that have unit truth values are listed below:

(0001)	AND	(0110)	EXCLUSIVE OR
(0111)	OR	(1001)	COINCIDENCE

Zero truth value

A zero truth value (z) is when X1o(z) = z for all values of X1. The constraints for a zero truth value of "0" and "1" are respectively:

z = 0: $0 + 0 = p = 0$

$1 + 0 = q = 0$

z = 1: $0 + 1 = q = 1$

$1 + 1 = r = 1$

For a zero truth value of either "0" or "1", the following compositions are satisfactory:

(0000)	ZERO	(0001)	AND
(1111)	ONE	(0111)	OR

Isomorphism

Isomorphism is a one to one transformation where the algebraic properties (commutativity, idempotency, etc.) are preserved. An isomorphic class is defined as the collection of all compositions which are isomorphic to a given composition and therefore to each other. A composition that is isomorphic to another composition is said to be its conjugate. In Boolean algebra, all that is necessary to find a composition's conjugate is to change 0's to 1's and 1's to 0's.

	0	1
0	0	1
1	1	1

	1	0
1	1	0
0	0	0

The above example shows that the OR and AND compositions are isomorphic. The other commutative isomorphic classes are listed below:

1) (0111) - (0001) OR-AND

2) (1000) - (1110) NOR-NAND

3) (0000) - (1111) ZERO-ONE

4) (0110) - (1001) EXCLUSIVE OR-COINCIDENCE

Distributivity

Distributivity is much more difficult to check than the other properties. In algebraic form, it is defined as $X1 \circ (X2 \oplus X3) = (X1 \circ X2) \oplus (X1 \circ X3)$, where

"o" and "⊕" denote different and arbitrary operators. An example that is quite familiar is the AND distributing over the OR, e.g., (X1) (X2 + X3) = (X1) (X2) + (X1) (X3). Other examples of distributivity are given below:

1) (X1) (X2 + X3) = (X1) (X2) + (X1) (X3) AND-OR

2) X1 + (X2) (X3) = (X1 + X2) (X1 + X3) OR-AND

3) (X1) (X2 ⊕ X3) = (X1) (X2) ⊕ (X1) (X3) AND-EXCLUSIVE OR

4) X1 + (X2 o X3) = (X1 + X2) o (X1 + X3) OR-COINCIDENCE

5. Functional Completeness

A. Binary

In order for a set of compositions to be functionally complete, it must be possible to generate all other compositions given only those compositions being tested. There are only two compositiions that are functionally complete within themselves, namely the NOR (1000) and the NAND (1110) compositions. From either one of these compositions, all other compositions can be derived. The NOR can be used as an illustration:

(1000) NOR (1000) = (0111)

Thus, by NORing the NOR composition with itself, the OR (0111) composition is formed. To review the algebra, the first elements of the two compositions, namely "1" and "1" are NORed giving "0". The second elements, "0" and "0" result in a "1", as do the third and fourth elements. Now, by NORing the OR composition and the NOR composition, the ZERO composition is formed:

(0111) NOR (1000) = (0000)

Several compositions can be used together in order to form a

functionally complete set. The AND (0001) and the NOT (1010) are two such compositions. Appendix I contains a computer program that will check a set of two-valued (binary) two-place functions for functional completeness. The AND and NOT compositions are used as an example and all 16 compositions are generated from these basic two. There is now a means of finding out what affect the algebraic properties discussed in the last section has on a set of compositions. Table I is a listing of the 16 two-place functions produced by the iterative operations of the AND and NOT compositions. The iterations are shown in the table for comparison purposes only and were easily obtained from the computer program output. It is pointed out that 42 operations were required to produce the 16 output compositions. Table II is also a listing of the 16 two-place functions, but in this table, they were produced by iterative operations of the NOR composition which is functional complete within itself. Note that now 88 operations were required to produce the 16 compositions with the NOR function. This is twice as many as for the AND/NOT set. The following conclusions can be drawn from the discussion in the last two sections:

(a) Before a set of compositions is useful in computer logic, (or any other logic for that matter), it must form a functionally complete algebra.

(b) In a mathematical sense, the most useful functionally complete set of compositions are those that have the most of the algebraic properties listed earlier. Since the AND composition is commutative, associative, and idempotent and has a unit

truth value and a zero truth value while the NOR composition is only commutative, the functionally complete set of compositions containing the AND composition is better than the functionally complete NOR composition since less devices would be required to build on arbitrary function,

(c) A composition need not have any of the algebraic properties to be useful. The NOT composition isn't even commutative, but it does have one useful and necessary characteristic. Notice that a "0" and "0" in the NOR composition gives a "1" output and that a "1" and a "1" gives a "0" output. Before a set of compositions is functionally complete, one of the set must have this characteristic. Since the AND composition does not invert a zero, some other composition must be included that does. The NOT composition has this characteristic and therefore is included even though it has no useful algebraic properties,

(d) Additional compositions often help make the set even more efficient. Table III is a list of the 16 two-place functions generated by the AND, NOT, and OR compositions. 28 operations are required with this set compared with 42 when only the AND and NOT were used. Thus, redundant compositions can be used to good advantage.

Although the expansion theorem will not be discussed until later, it is worth mentioning at this time that all of the above comments can be

put into more concise language by stating that the simpler the expansion theorem the better.

Everything that has been discussed thus far carries over into the ternary situation as well.

B. Ternary

The test for functional completeness of ternary devices is somewhat more complicated than for binary devices. In 1939, Slupecki proved that a collection of compositions is functionally complete in n-valued logic (n greater than two) if and only if the following two conditions are satisfied:

(1). It must be possible to express all one-valued functions in terms of these compositions.

(2). It must be possible to express at least one particular two-place function F(a,b) in terms of these compositions for which the following is true:

a). For every truth value, i, in the system, there exists a pair of truth values j and k such that F(j,k) = i.

b). A set of truth values (a,b,c,d) exists such that $F(a,c) \neq F(a,d)$, $F(a,c) \neq F(b,c)$, and $F(a,d) \neq F(b,c)$.

In Ref. [10] , the saturable Hall element is discussed and the resulting compositions (0,0,1,1,2,2) and (2,2,1,1,0,0) are shown to be functionally complete. Only six digits are used to represent compositions which are commutative, i.e., where F(0,1) = F(1,0), F(0,2) = F(2,0), and F(1,2) = F(2,1). The second condition stated above is easily satisfied as follows:

a). F(j,k) = i for (0,0,1,1,2,2)

$F(0,1) = 0$ $F(0,2) = 1$ $F(1,2) = 2$

Actually, either of the compositions would satisfy this requirement. The same holds true for the requirement of (2b), so the first composition, (0,0,1,1,2,2) is used:

b. Letting a = 2, b = 0, c = 0, and d = 2:

$F(a,c) \neq F(a,d)$

$F(2,0) = 1 \neq F(2,2) = 2$

$F(a,c) \neq F(b,c)$

$F(2,0) = 1 \neq F(0,0) = 0$

$F(a,d) \neq F(b,c)$

$F(2,2) = 2 \neq F(0,0) = 0$

The first condition can be satisfied only by physically trying to construct the 27 one-place functions by use of the given compositions. Certain obvious requirements have to be met before the 27 functions can be generated. For example, at least one of the compositions must have the characteristic that a "0" and a "0" gives an output other than zero. The same holds true for a "1" and "1", and a "2" and "2". Other conditions could be specified that would insure the constructability of the 27 functions. However, Appendix II contains a computer program that attempts to generate these functions by a straight brute force method. If one were checking the functional completeness of only one particular set of compositions, it would naturally not be advantageous to write a computer program for this one trial. But if a check is being made on a series of compositions, the time saved once the program is written

would definitely be worth while.

6. An Experimental Ternary Adder

There are many compositions that are functionally complete. Swift [3] and Martin [4] have shown that there are 3774 three-valued Sheffer functions (functions that are functionally complete by themselves). It is easy to imagine that there are many more when several compositions are combined. But, building devices that behave like these functions is not always possible. In the binary case, no simple device has yet been built that has the characteristics of the NOR or NAND functions. Even when such a device is available, it may be necessary to combine many such devices in a complex manner to accomplish a particular task. In fact, this is probably one of the basic reasons why an actual computer has not been built that operates in ternary.

Several devices have been proposed that seem to be approaching the point where they may be useful. In some cases these can be combined with existing devices to create an even more useful set. But, it is possible that certain properties may make two devices incompatible, such as different voltage levels.

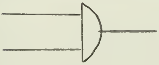
An attempt was made here to find a functionally complete set of compositions by the use of diodes and transistors. It seems that the inverter action of a transistor and the AND and OR action of the diode could be utilized in ternary in a fashion similar to binary to create a usable system.

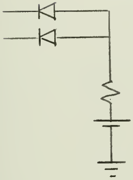
The Zener diode was added because its characteristics were found to be quite useful. There are four compositions in the set. The AND and

the OR gates are exactly the same as for binary; the AND output assumes the lowest value of the inputs and the OR output assumes the highest value of the inputs.

AND

$Z = XY$

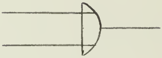


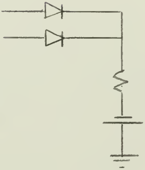


	0	1	2
0	0	0	0
1	0	1	1
2	0	1	2

OR

$Z = X + Y$



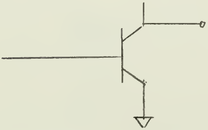


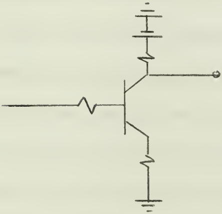
	0	1	2
0	0	1	2
1	1	1	2
2	2	2	2

The transistor is also used in a manner similar to binary. It is quite easy to demonstrate one method in which the transistor can be used. Let "0" be zero volts, "1" be five volts and "2" be ten volts. If the parameters are picked that will make zero volts input have the transistor

cutoff to give ten volts output and three volts input drive the transistor into saturation to give zero volts output, then a "0" in will give a "2" out and a "1" or a "2" in will give a "0" out. Depending on how the parameters are picked there are several possible compositions that can be formed. The inverter device is represented as below:

INVERTER
(INV X)





0	2
1	0
2	0

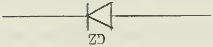
A Zener diode with five volt breakdown would allow zero volts output for zero or five volts input, and five volts output for ten volts input. This

[illegible]

[illegible]

device could be called a limiter because it limits the amount of voltage at the output.

LIMITER
(LIM X)



0	0
1	0
2	1

These four compositions when used in conjunction with a constant "1", have proven to be a functionally complete set. The computer program included in the appendix on three-valued devices shows how the 27 one-place functions were derived from this set. For the second requirement set forth by Zupecki, either the OR or the AND compositions have all three values listed as outputs (condition 2(a)) and letting $a = 0$, $b = 2$, $c = 1$, $d = 0$, the OR compositions can be used to satisfy condition 2(b).

$F(a,c) \neq F(a,d)$

$F(0,1) = 1 \neq F(0,0) = 0$

$F(a,c) \neq F(b,c)$

$F(0,1) = 1 \neq F(2,1) = 2$

$F(a,d) \neq F(b,c)$

$F(0,0) = 0 \neq F(2,1) = 2$

This proves that the set is functionally complete.

To see just how useful this set is, a ternary half adder, full adder and carry were designed and the design compared to other suggested designs. The procedure used in the design is similar to the "sums of products" in binary design. First, the desired end result for the half adder could be expressed as the AND product of six other compositions, where each possible commutative output is derived separately.

$$\begin{array}{|ccc}\hline 0&1&2\\ 1&2&0\\ 2&0&1\end{array} = \begin{array}{|ccc}\hline 0&2&2\\ 2&2&2\\ 2&2&2\end{array} \circ \begin{array}{|ccc}\hline 2&1&2\\ 1&2&2\\ 2&2&2\end{array} \circ \begin{array}{|ccc}\hline 2&2&2\\ 2&2&2\\ 2&2&2\end{array} \circ$$

$$\begin{array}{|ccc}\hline 2&2&2\\ 2&2&2\\ 2&2&2\end{array} \circ \begin{array}{|ccc}\hline 2&2&2\\ 2&2&0\\ 2&0&2\end{array} \circ \begin{array}{|ccc}\hline 2&2&2\\ 2&2&2\\ 2&2&1\end{array}$$

The constant "2" outputs are redundant and can be ignored. Combining the first two compositions into one composition and the last two into one simplifies the construction.

$$\begin{array}{|ccc}\hline 0&1&2\\ 1&2&0\\ 2&0&1\end{array} = \begin{array}{|ccc}\hline 0&1&2\\ 1&2&2\\ 2&2&2\end{array} \circ \begin{array}{|ccc}\hline 2&2&2\\ 2&2&0\\ 2&0&1\end{array}$$

These two compositions are constructed as follows:

$$\begin{array}{|ccc}\hline 0&1&2\\ 1&2&2\\ 2&2&2\end{array} = \text{(X1 OR X2) OR (INV(INV(X1 AND X2)))}$$

= (012122) + (INV(INV(000112)))

= (012122) + (INV(222000)

= (012122) + (000222)

= (012222)

2 2 2 = (INV(LIM(X1 OR X2))) OR (LIM(X1 AND X2))

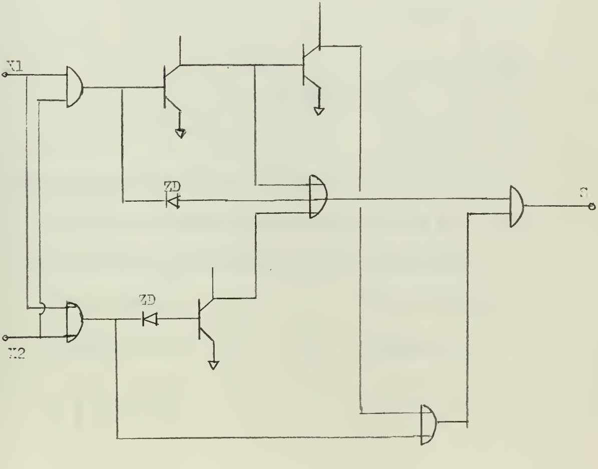
2 2 0 OR (INV(X1 AND X2))

2 0 1 = (INV(LIM(012122))) + (LIM(000112)) + (INV(000112))

= (INV(001011)) + (000001) + (222000)

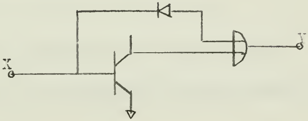
= (220200) + (000001) + (222000)
= (222201)

A diagram of this circuit using the symbols defined earlier is given below:

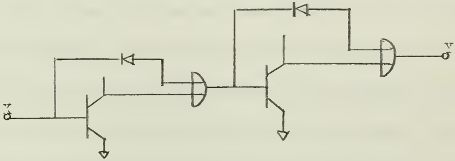


For the design of a full adder, one simple minded approach is to add one to the half adder output when the previous carry equals one and to add zero when the previous carry is zero. There are several approaches possible in adding one. The method chosen here was to subtract one twice.

X	Y
0	2
1	0
2	1



X	Y
0	1
1	2
2	0



The previous carry acts as a gate and can be defined as

$$Z(\text{output}) = (\bar{C} \text{ AND } (X \text{ plus } Y)) \text{ OR } (C \text{ AND } (X \text{ plus } Y \text{ plus } 1))$$

The next carry is generated by the two compositions:

When previous carry = 0

	0	1	2
0	0	0	0
1	0	0	1
2	0	1	1

When previous carry = 1

	0	1	2
0	0	0	1
1	0	1	1
2	1	1	1

Carry = ($\overline{C}$ AND ((X1 AND X2) AND (LIM(X1 OR X2)))) OR

(C AND ((LIM(X1 OR X2)) OR (LIM(X1 plus X2))))

= ($\overline{C}$ AND ((000112) AND (LIM(012122)))) OR

(C AND ((LIM(012122)) OR (LIM(012201))))

= ($\overline{C}$ AND ((000112) AND (001011))) OR

(C AND ((001011) OR (001100)))

= ($\overline{C}$ AND (000011)) OR (C AND (001111))

The logic circuit for the ternary full adder is shown in Fig. (3).

7. Expansion Theorems

The design of the half and full adders just described was done on an intuitive basis. This procedure would not be satisfactory if more complicated circuits were required. For every functionally complete set of compositions there exists an expansion theorem. An expansion theorem relates a function of N variables to a function of N-1 variables. The two-valued Boolean expansion theorem, f(X1,X2,..XM) = (X1)f(1,X2,..XM) + ($\overline{X1}$)f(0,X2,...,XM) is well known.

X1

		0	1		
	0	0	0	0	
	1	1	0	0	
X2					X3
	0	0	1	1	
	1	0	0	1	

In the example given above, f(X1,X2,X3) = $\overline{X1}X2\overline{X3} + X1\overline{X2}X3$. From

the theorem, we have f(X1,X2,X3) = (X1)f(1,X2,X3) + $(\overline{X1})$f(0,X2,X3). For the output when X1, X2, X3 = 0, then f(0,0,0) = (0)f(1,0,0) + (1)f(0,0,0) = f(0,0,0) as is desired. For the output when X1 = 1, X2 and X3 = 0, then f(1,0,0) = (1)f(1,0,0) + (0)f(0,0,0) = f(1,0,0). Thus, if the value of X1 is known, the expansion theorem simplifies the original function of three variables into a function of two variables. These functions could then be reduced even further into functions of one variable if desired.

In the ternary case, a somewhat similar formula can be made:

		X1		
		0	1	2
	0	0	1	2
X2	1	1	2	0
	2	2	0	1

For the example truth table above, it is noticed that when X2 equals zero, a permutation; "0" to "0", "1" to "1", and "2" to "2" of X1 takes place. Also, when X2 is one, the permutation is "0" to "1", "1" to "2", and "2" to "0", and when X2 is two, the permutation is "0" to "2", "1" to "0", and "2" to "1". One method of writing a formula for this particular composition would be to write in general, f(X1,X2) = g1(X2)f(0,X1) + g2(X2)f(1,X1) + g3(X2)f(2,X1). This is explained as follows: If the X2 input is zero, then it is desired that g1(X2) = g1(0) equal two and g2(0) and g3(0) equal zero. At the same time, it is desired that f(0,X1) gives the first permutation ("0" to "0", "1" to "1", "2" to "2"). In equation

form then, it is desired that when X2 equals zero, f(0,X1) = (2)f(0,X1) + (0)f(1,X1) + (0)f(2,X1) = (2)f(0,X1). For X2 equal to one, the first and last factors must equal zero, thus g1(1) and g2(1) must be constructed with this in mind. The second factor must insure the permutation "0" to "1", "1" to "2", and "2" to "0" and thus g2(1) should be made equal to two. When X2 equals two the same type of procedure holds true and g1(2) and g2(2) should equal zero while g3(2) should equal two while f(2,X1) is constructed to do the third permutation.

g1(X2), g2(X2), and g3(X2) were determined with these restrictions in mind from the available compositions and matched to the correct permutation.

g1(X2) = (INV(X2))

g2(X2) = (INV(INV(X2))) AND (INV(LIM(X2)))

g3(X2) = (INV(INV(LIM(X2))))

f(0,X1) = (X1)

f(1,X1) = (INV(INV(X1)OR LIM(X1))) OR (LIM(INV(X1) OR LIM(X1)))

f(2,X1) = (INV(X1)) OR (LIM(X1))

f(X1,X2) = ((INV(X2)) AND (X1)) OR
((INV(INV(X2))) AND (INV(LIM(X2))) AND (INV(INV(X1)
OR LIM(X1))) OR (LIM(INV(X1) OR LIM(X1)))) OR
((INV(INV(LIM(X2)))) AND (INV(X1))AND (LIM(X1)))

This still is not an expansion theorem for the four compositions, but it is a good example of how an expansion theorem is constructed and this particular composition will also be of use later in the section. The half adder just constructed is only slightly more complicated than the one

constructed on an intuitive basis. This one contains five inverters, three limiters and eight OR/AND gates compared to three inverters, two limiters and five OR/AND gates.

The truth table for the full adder sum is given below. The same type of permutations take place on X1, but now, which permutation is used depends upon two variables, X2 and C. Permutation number one, (f(0,X1)), now takes place when X2 and C are both zero or when X2 equals two and C equals one.

	X1			
X2	0	1	2	C
0	0	1	2	0
0	1	2	0	1
1	1	2	0	0
1	2	0	1	1
2	2	0	1	0
2	0	1	2	1

The full adder thus has the formula:

f(X1,X2,C) = (((INV(C)) AND (INV(INV(X2)))) OR ((INV(INV(C))) AND
(INV(X2)))) AND ((INV(INV(X1) OR LIM(X1))) OR
(LIM(INV(X1) OR LIM(X1)))) OR
(((INV(INV(X2))) AND (INV(INV(C))) AND (INV(LIM(X2))))
OR ((INV(C) AND (INV(INV(LIM(X2)))))) AND
(INV(X1) OR LIM(X1))) OR

((((INV(INV(LIM(X2))) AND (X2) AND (INV(INV(C)))) OR

((INV(X2)) AND (INV(C)))) AND (X1))

For comparison purposes, this full adder not including the next carry would contain eight inverters, three limiters, and sixteen OR/AND gates compared to seven inverters, four limiters, and nine OR/AND gates for the full adder described earlier.

With the procedures just used in constructing the half and full adders, the actual derivation of the expansion theorem is not too complicated. For the derivation of the expansion theorem, six functions are defined:

(1) 000 = h0(a) = h0 = LIM(LIM(a))

(2) 111 = h1(a) = h1 = 1

(3) 222 = h2(a) = h2 = INV(LIM(LIM(a)))

(4) 200 = g0(a) = INV(a)

(5) 020 = g1(a) = INV(LIM(a)) AND INV(INV(a))

(6) 002 = g2(a) = INV(INV(LIM(a)))

The first three functions are constants. Whatever the value of a, the input, the output is constant. These constants would normally be available in the form of voltage sources. The next three functions are permutations of the input a and have the characteristic that only one value of a will give a two output while any other value of a will give a zero output. As an example g0(a) is the permutation, "0" to "2", "1" to "0", and "2" to "0". Any one-place function can be expressed in terms of these given functions by the theorem:

f(x) = (g0(X) AND h(f(0))) OR (g1(X) AND h(f(1))) OR (g2(X) AND h(f(2)))

For example, to obtain the one-place function (210), then:

f(0) = (g0(0) AND h(f(0))) OR (g1(0) AND h(f(1))) OR (g2(0) AND h(f(2)))

= ((2) AND h(2)) OR ((0) AND h(1)) OR ((0) AND h(0))

= (2) AND (2)

= 2

f(1) = ((0) AND h(2)) OR ((1) AND h(1)) OR ((0) AND h(0))

= (1) AND (1)

= 1

f(2) = ((0) AND h(2)) OR ((0) AND h(1)) OR ((2) AND h(0))

= (2) AND (0)

= 0

The general expansion theorem for the four compositions under discussion can now be written down directly:

f(X1,X2,...,XM) = (g0(X1) AND f(0,X2,X3,...,XM)) OR
(g1(X1) AND f(1,X2,X3,...,XM)) OR
(g2(X1) AND f(2,X2,X3,...,XM))

This expansion theorem is very similar to the Boolean expansion theorem as one might expect because it was derived in almost exactly the same fashion. In fact, Post [1] developed an expansion theorem for a set of three compositions; the AND and the OR as defined here, and a third called the SUCCESSOR OF X. The SUCCESSOR OF X is the permutation of "0" to "1", "1" to "2", and "2" to "0". Since the SUCCESSOR OF X is obtained from a combination of several of the compositions used above, (SUCCESSOR OF X = INV(INV(X) OR LIM(X)) OR LIM(INV(X) OR LIM(X))), then

there can expect to be a close similarity between the two sets. Post's algebra holds for any N-valued system and Rosenbloom [2] showed that for N = 2, the Post algebra degenerates into the Boolean algebra. Thus, it is apparent that the similarity between the Boolean expansion theorem and the one developed above is not purely coincidence. If the set of compositions under study do not contain the AND and OR compositions, the similarity ends although the methods used are still perfectly general.

The full adder that was developed using permutations can now be explained as one particular example of the use of the expansion theorem. In simplified form, the full adder equation is:

f(X1,X2,C) = (f1(X2,C)) AND (Permutation #1) OR
(f2(X2,C)) AND (Permutation #2) OR
(f3(X2,C)) AND (Permutation #3)

f(X2,C) represents g(X) and Permutation (X) represents f(X1,X2=a, C=b) where a function of three variables is being related to a function of one variable. Thus, in developing the full adder, two steps were taken at once. In the more general case, one would say that, for example, when X2 equal zero, then the permutation of X1 would depend upon the value of C. This would give a function of three variables expressed in relation to a function of two variables. The end result could be synthesized to the same full adder.

8. Experimental Results

To prove the compatibility of the four compositions, an experimental half adder using actual transistors and diodes was built. Zero volts was

used for a "0", five volts for a "1", and ten volts for a "2". The simplest possible design was used and no attempt was made to optimize such things as power supply voltages, transistor types, resistance values, etc. The circuit used is given in Fig. (4) and the desired and actual results are given below in truth table form.

Desired

	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

	0	5	10
0	0	5	10
5	5	10	0
10	10	0	5

Actual

	0	5	10
0	2.4	4.8	8.2
5	4.8	10.0	2.8
10	8.2	2.8	5.0

If the voltages actually measured were now rounded off in the range, zero equals zero to three, five equals three to seven, and ten equals seven to ten, then the circuit can be considered to work satisfactory, thus proving the feasibility of this set of compositions.

The voltages were not exactly as desired for two primary reasons. First, the Zener diodes used were of low quality with a fairly low forward resistance. When connected in series with the base resistance of the inverter which was approximately 13K ohms, the Zener diode did not develop its full voltage drop as desired. Second, the circuit was inter-

connected in such a way that various feedback loops existed which prevented the transistors from being fully cut-off of from being in complete saturation. These two problems could be dealt with, with an actual circuit of higher quality Zener diodes and by either allowing for the feedback or preventing it from affecting the results. Since the object of this experiment was to prove the compatibility of the devices, it was not considered necessary to try and correct the above deficiencies.

There are two types of comparison possible on the set of compositions studied. One can compare the complexity of the adder circuits to other suggested adder circuits, or one can compare the complexity of the expansion theorems of various sets of compositions. Each will be dealt with in turn.

A comparison of the half adder built here and one proposed by Vacca [6] shows that the half adder here would require three transistors and 12 diodes compared with five inverters and 12 diodes. There is a slight improvement here plus the fact that Vacca proposed three different inverter configurations compared with one here.

A comparison of the full adder developed here and one proposed by Vacca shows that the full adder here would require seven transistors and 36 diodes compared with 14 transistors and 30 diodes. Since the cost of construction of the inverter devices would be quite large compared to the OR/AND gates, there is a good improvement in design here.

A comparison of the full adder developed here and one proposed by Lowenschuss [5] using Rutz transistors shows that the full adder here

would require 27 devices compared to nine for the Rutz adder. This is quite misleading since each of the nine devices used by Lowenschuss requires two two-collector transistors. With the cost of construction considered, it could be expected that the 27 device adder may very well be less expensive than the nine device adder. It can not be ignored that the nine device adder would still be a less complicated circuit.

A method of comparison of expansion theorem's complexity has been developed by Shannon and Lowenschuss [5] which is a better indication of the usefulness of a set of compositions than comparing the construction of adder circuits. For three-valued algebras, the upper bound on the number of switching devices required to implement an arbitrary N-place function is given by the formula,

$$s(n) = (3^{(n-1)}(B + 2s(1)) - B)/2$$

$s(n)$ = upper bound on number of switching devices required to implement an arbitrary N-place function

$s(1)$ = upper bound on number of switching devices required to implement an arbitrary one-place function

B = number of compositions in the expansion theorem

The best expansion theorem thus far found is for the Modular Algebra which has two compositions, the base-three sum without carry, and the base-three product without carry. No actual device has yet been discovered

that has these characteristics. It is interesting to note that the base-three summer is the half adder output that was constructed with transistors and diodes.

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

x	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

The upper limit on the number of devices in this case has been determined by Lowenschuss to be,

$$s(n) = 2(3^n - 1)$$

For the Rutz transistor compositions proposed by Lowenschuss, the upper limit is

$$\begin{aligned} s(n) &= 18(3^{n-1}) - 12 \\ &= 6(3^n - 2) \end{aligned}$$

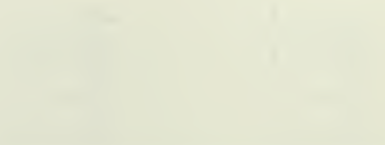
For the saturable Hall logic element discussed by Kier [10], the upper limit is:

$$\begin{aligned} s(n) &= (3^{n-1}(13 + 2(8)) - 13)/2 \\ &= (9(3^n) - 13)/2 \\ &= 4(3^n) - 6 \end{aligned}$$

For the four compositions used above, the most difficult one-place functions to obtain are (021), (120), and (102). For each of these, six operations are required. Assuming that the constants "0", "1", and "2" are available as sources, there are ten compositions in the expansion theorem. Therefore, B = 10 and s(1) = six:

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

$$\begin{aligned} s(n) &= (3^{n-1}(10 + 2(6)) - 10)/2 \\ &= (3^{n-1}(22) - 10)/2 \\ &= (3^{n-1}(11) - 5) \\ &= 3(3^{n}) - 5 \end{aligned}$$

The upper limit on the number of devices required to generate an arbitrary function for several values of n are tabulated below for the various expansion theorems:

n	Modular	Rutz	Hall	Post(the set proposed herein)
1	4	6	6	4
2	16	42	30	22
3	52	150	102	76
4	160	484	318	238

In the above calculations of s(n), numbers were rounded off, therefore, the upper limit on the number of devices is not exactly correct. But, it can be seen that the set of four compositions proposed here does well for itself. Also worthy of noting is, that this is an upper limit on the number of devices and not an absolute minimum. For example, when n equals two, the upper limit on the half adder proposed here is 22, whereas only 10 devices were actually required.

9. Conclusion

The set of four compositions introduced here has the following characteristics:

(a) It is a functionally complete set with a fairly simple expansion theorem,

(b) The AND and the OR compositions have algebraic properties of associativity, idempotency, and commutativity, and they have a unit truth value and a zero truth value. This indicates that they would be very useful in the manipulation of equations,

(c) The three elements required, transistor, Zener diode, and regular diode, are compatible for hardware installation,

(d) The algebra used is closely related to Boolean algebra and as such would have a closer relationship to present day computer design than most other algebras,

(e) With this close relationship to Boolean algebra, it might be possible to interlace binary and ternary logic. At least, the interface problem would be minimized.

There is still one major disadvantage to ternary logic. Ledley [8] shows a binary full adder made up of one inverter and eight AND/OR gates. With the advantage that ternary has over binary in digit space, one could allow for a more complex ternary circuit, but the seven inverters and 36 diodes is much too complex. This could be considered a drawback rather than a disadvantage. It could be overcome by two methods. The first is the possibility of finding a suitable minimization procedure. The second is by using additional redundant compositions. One composition that appears to have useful properties and that could be built with a d.c. amplifier arrangement, is the following:

0	0
1	2
2	2

Another composition that would be useful and that would make a true Post Algebra, although no simple transistor circuit has been devised for it is the following:

0	1
1	2
2	0

Still a third suggested composition might be the following:

0	0
1	2
2	0

Any or all of these could be used to reduce the complexity of the adder circuit. One might argue that too many compositions are being introduced for the set to be useful, but, this is similar to the binary situation where the AND and NOT compositions are functionally complete by themselves and the OR composition is added just for simplification purposes.

Although the major portion of the discussion herein has been directed toward a particular set of compositions that are related closely to Boolean Algebra, there was no intention of underemphasizing the possible use of other sets of compositions. The Modular Sum and Product Algebra has a major advantage in its close relation to normal arithmetic. The saturable Hall logic element discussed by Keir is particularly suitable for computer design.

Although no mention was made of uses other than for adder circuits, almost everything that was stated holds true for other operations. Multiplication by two in binary can be accomplished by a shift register and multiplication by three in ternary can be accomplished by a shift register also. Having three stable states might simplify input/output; the existence of a zero on an input/output line might be used to indicate "ready to read", a one might indicate "ready to write", and a two might indicate "wait". Ternary also leads to the possibility of a built-in "IF" instruction where greater than, less than, or equal to could all be done in one instruction.

It does not appear appropriate to predict if a ternary computer will ever be put into operation. With the advent of microminimization, it may be a long time before the need to use some other base system exists. Once a need has been generated, there will still be many problems the least of which may be the finding of useful devices. Other major problems that will have to be considered include interface between binary and N-nary computers and the teaching of the new algebra to engineers. However, most research in the field of ternary logic is extendable to N-valued logic and therefore will be useful to any base system studied in the future.

FIGURES AND TABLES

Fig. 1

Binary and Ternary Truth Tables of Four Inputs

Binary truth table of four inputs

X1

		0	1	0	1		
	0	0	0	0	0	0	
X2	1	0	0	0	0	0	X4
	0	1	0	1	0	1	
	1	0	0	0	0	1	
		0	0	1	1		

X2

$$\text{Output} = (\overline{X1})(\overline{X2})(\overline{X3})(X4) + (\overline{X1})(\overline{X2})(X3)(X4)$$

Ternary truth table of four inputs

	A1	A1	A1	A2	A2	A2	A3	A3	A3	
B1	0	0	0	0	0	0	0	0	0	D1
B1	0	0	2	0	0	0	0	0	0	D2
B1	0	0	0	1	2	0	0	0	0	D3
B2	0	0	0	0	0	0	0	0	0	D1
B2	1	1	0	0	0	0	0	0	0	D2
B2	0	0	0	0	0	0	0	0	0	D3
B3	0	0	0	0	0	0	0	0	0	D1
B3	0	0	0	0	0	0	0	0	0	D2
B3	0	0	0	0	0	1	2	0	0	D3
	C1	C2	C3	C1	C2	C3	C1	C2	C3	

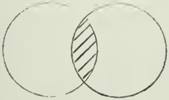
"2's" output = A1B1C3D2 + A2B1C2D3 + A3B3C1D3

"1's" output = A2B1C1D3 + A1B2C1D2 + A1B2C2D2 + A2B3C3D3

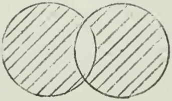
Fig. 2

Venn Diagrams for Binary and Ternary

Binary

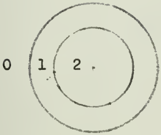


	0	1
0	0	0
1	0	1



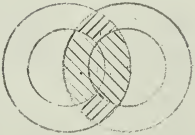
	0	1
0	0	1
1	1	0

Ternary

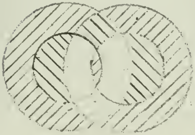


▨ = "1" output

▧ = "2" output



	0	1	2
0	0	0	0
1	0	1	2
2	0	2	2



	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Fig. 3

Logic Diagram for Ternary Full Adder

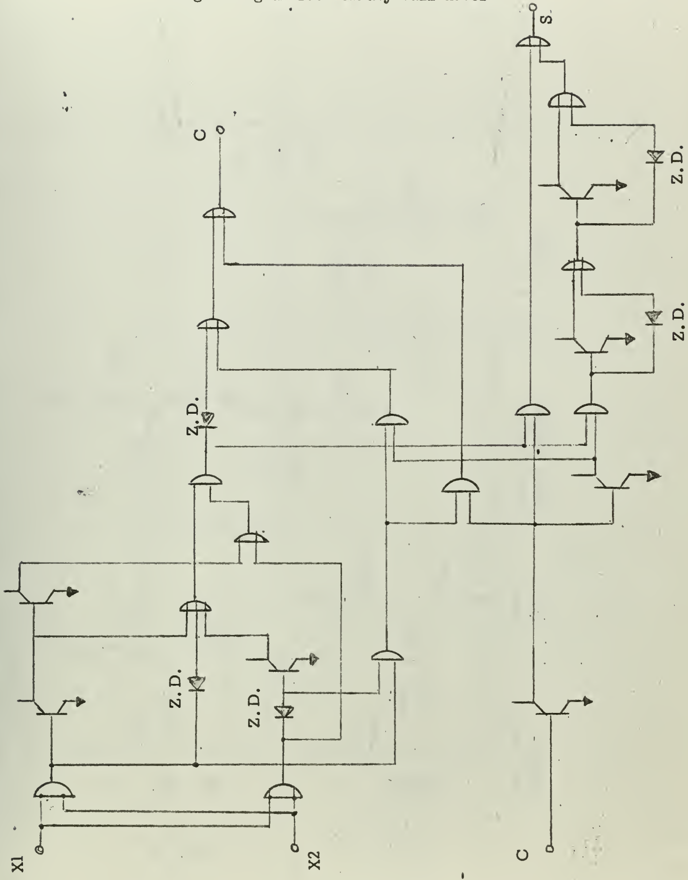


Fig. 4

Circuit Diagram for Ternary Half Adder

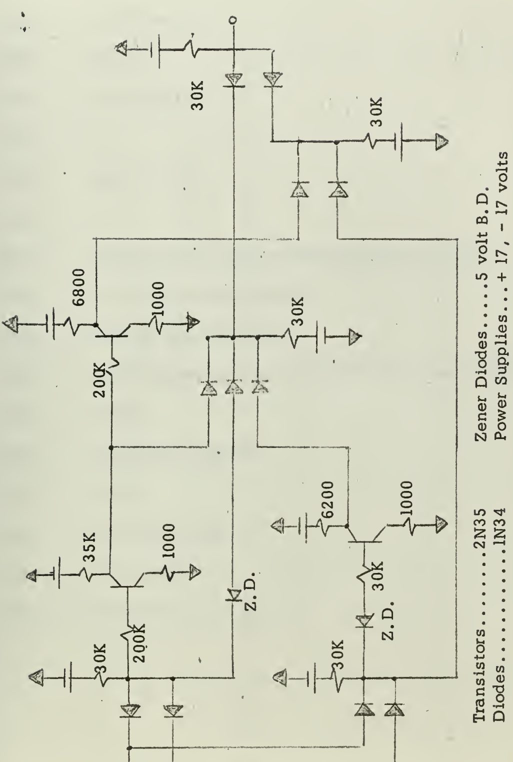


Table I

Functional Completeness of AND/OR Compositions

0000.....NOT(X) AND (X)

0001.....X AND Y

0010.....NOT(X) AND Y

0011.....Y

0100.....NOT(Y) AND X

0101.....X

0110.....(NOT(X AND Y)) AND (NOT(NOT(X) AND NOT(Y)))

0111.....NOT(NOT(X) AND NOT(Y))

1000.....(NOT(X)) AND (NOT(Y))

1001.....NOT((NOT(X AND Y)) AND (NOT(NOT(X) AND NOT(Y))))

1010.....NOT(X)

1011.....NOT(NOT(Y) AND X)

1100.....NOT(Y)

1101.....NOT(NOT(X) AND Y)

1110.....NOT(X AND Y)

1111.....NOT(NOT(X) AND (X))

Table II

Functional Completeness of NOR Composition

0000.....(X NOR Y) NOR ((X NOR Y) NOR (X NOR Y))

0001.....((X NOR Y) NOR (X)) NOR (((X NOR Y) NOR ((X NOR Y) NOR (X NOR Y))) NOR (Y))

0010.....(X NOR Y) NOR (Y)

0011.....Y

0100.....(X NOR Y) NOR (X)

0101.....X

0110.....((((X NOR Y) NOR (X)) NOR ((X NOR Y) NOR (Y)))) NOR (X NOR Y)

0111.....(X NOR Y) NOR (X NOR Y)

1000.....X NOR Y

1001.....((X NOR Y) NOR (X)) NOR ((X NOR Y) NOR (Y))

1010.....((X NOR Y) NOR ((X NOR Y) NOR (X NOR Y))) NOR (X)

1011.....((X NOR Y) NOR ((X NOR Y) NOR (X NOR Y))) NOR ((X NOR Y) NOR (Y))

1100.....((X NOR Y) NOR ((X NOR Y) NOR (X NOR Y))) NOR (Y)

1101.....((X NOR Y) NOR ((X NOR Y) NOR (X NOR Y))) NOR ((X NOR Y) NOR (X))

1110.....(((X NOR Y) NOR (X)) NOR (((X NOR Y) NOR ((X NOR Y) NOR (X NOR Y))) NOR (Y))) NOR ((X NOR Y) NOR ((X NOR Y) NOR (X NOR Y)))

1111.....((X NOR Y) NOR ((X NOR Y) NOR (X NOR Y))) NOR ((X NOR Y) NOR ((X NOR Y) NOR (X NOR Y)))

Table III

Functional Completeness of AND/OR/NOT Compositions

0000.....NOT(X) AND X

0001.....X AND Y

0010.....NOT(X) AND Y

0011.....Y

0100.....NOT(Y) AND X

0101.....X

0110.....(X OR Y) AND (NOT(X AND Y))

0111.....X OR Y

1000.....NOT(X OR Y)

1001.....(NOT(X OR Y)) OR (X AND Y)

1010.....NOT(X)

1011.....NOT(X) OR Y

1100.....NOT(Y)

1101.....NOT(Y) OR X

1110.....NOT(X AND Y)

1111.....NOT(X) OR X

BIBLIOGRAPHY

1. Post, E. L. Introduction to a general theory of elementary propositions. American Journal of Mathematics, v. 43, 1921: 163-185.

2. Rosenbloom, P. L. Post algebras I postulates and general theory. American Journal of Mathematics, v. 64, 1942: 167-188.

3. Swift, J. D. Algebraic properties of N-valued propositional calculi. American Mathematics Monthly, v. 59, 1952: 612-621.

4. Martin, N. M. The sheffer functions of 3-valued logic. Journal of Symbolic Logic, v. 19, 1954: 45-51.

5. Lowenschuss, O. Non-binary switching theory. 1958 IRE National Convention Record, part 4, 1958: 305-317.

6. Vacca, R. A three valued system of logic and its applications to base three digital circuits. Proceedings of International Conference on Information Processing, UNESCO House, Paris, France, 1959.

7. Scott, N. R. Analog and digital computer technology, McGraw-Hill, 1960.

8. Ledley, R. S. Digital computer control engineering, McGraw-Hill, 1960.

9. Lockheed Missles and Space Company Electronics Sciences Laboratory. On the applicability of ternary algebras to combinational switching network synthesis, by Y. A. Keir. May, 1963. Research report 8-44-63-4; AF 33(657)-8777.

10. Lockheed Missiles and Space Company Electronics Sciences Laboratory. The functional capability of the saturable Hall logic element, by Y. A. Keir. 7 August 1963. RACE Note E.44.

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

APPENDICES

APPENDIX I

Derivation of Two-place Two-valued Functions

The computer program that follows checks to see if a set of compositions is functionally complete. The method used is similar to the way the OR and ZERO compositions were generated in the section on functional completeness. It takes two given or generated compositions forms a new composition by use of one of the desired operations. It then checks to see if the newly formed composition is unique from those already generated.

In addition, the program computes a comparative cost for constructing a particular composition if a cost factor for each of the compositions being tested is included.

The output not only gives a direct answer as to whether the given compositions are functionally complete, but also can be used to determine exactly how each new composition is generated. By noting the values of "i" and "j", the two compositions used, Y(i) and Y(j) can be determined and by noting the value of "L", it can be determined which operation was performed on these compositions.

The results of testing the AND and NOT set of compositions is included and can be used to illustrate the above. In testing the AND/NOT compositions, four input compositions (NCOMPS) are used, (0000), (1010), (1100), and (0001). The ZERO composition is included for ease of programming. The (1010) composition is NOT X1, the (1100) composition is NOT X2, and the (0001) composition is X1 AND X2. Three actual opera-

tions (NFUNCS) are used. The A1(L) through A4(L) give the desired results from these operations. A1(1) represents the desired result of two inputs, "0" and "0". In the case of the AND, A1(1) is a "0". A2(1) is the result of a "0" and "1", A3(1) is the result of a "1" and "0", and A4(1) is the result of a "1" and "1". Thus for the AND operation, A1(1), A2(1), A3(1), and A4(1) are respectively, "0", "0", "0", and "1". The operation for L equal to two is the NOT X1 and gives A1(2) through A4(2) values of "1", "0", "1", and "0" respectively. Similarly, NOT X2 gives values of "1", "1", "0", and "0" respectively for A1(3) through A4(3).

From the printout for the AND/NOT test, the first new composition to be formed was obtained with "i" = 1, "j" = 1, and "L" = 2. Noting that Y(i) = Y(1) = (1010) and Y(j) = Y(1) = (1010), and that L = 2 indicates that the second operation (1100) was performed, the new composition can be reconstructed. The first elements, "1" and "1", result in a "0", as do the third elements. The second and fourth elements of "0" and "0" result in a "1". The new composition is (0101) which checks with the result printed as Y(5). Continuing to the second new composition, "L" = 1 indicates that the AND operation (0001) was performed on Y(i) = Y(1) = (1010) and Y(j) = Y(2) = (1100).

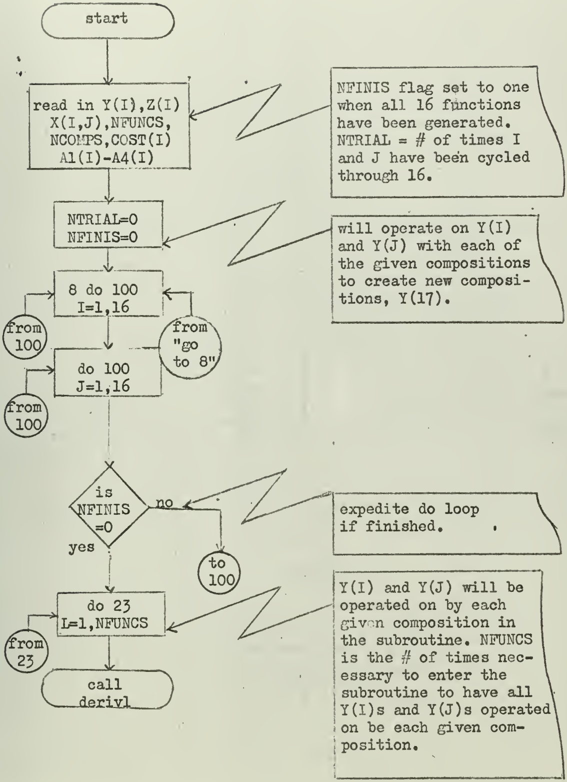
(1010)		0	1	(1100) = (1000)
	0	0	0	
	1	0	1	

Thus, the (1000) composition is formed and printed as Y(6). All 16 compositions are generated and a final printout is made. In this example, $2.00 was used as a basis of cost for an AND operation where two diodes

are necessary, and \$2.00 was used for the NOT operations where one transistor is necessary. The cost of constructing Y(5) is the cost of Y(1) plus the cost of Y(1) plus the cost of the operation. This gives a total of \$2.00 + \$2.00 + \$2.00 = \$6.00.

Appendix I, Fig. 1

Flow Chart for Deriving Two-valued Two-place Functions



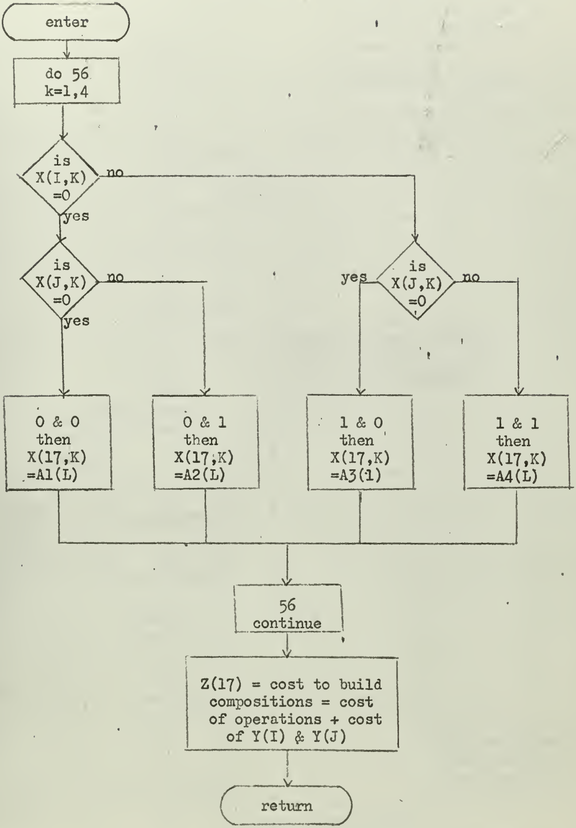
NFINIS flag set to one when all 16 functions have been generated. NTRIAL = # of times I and J have been cycled through 16.

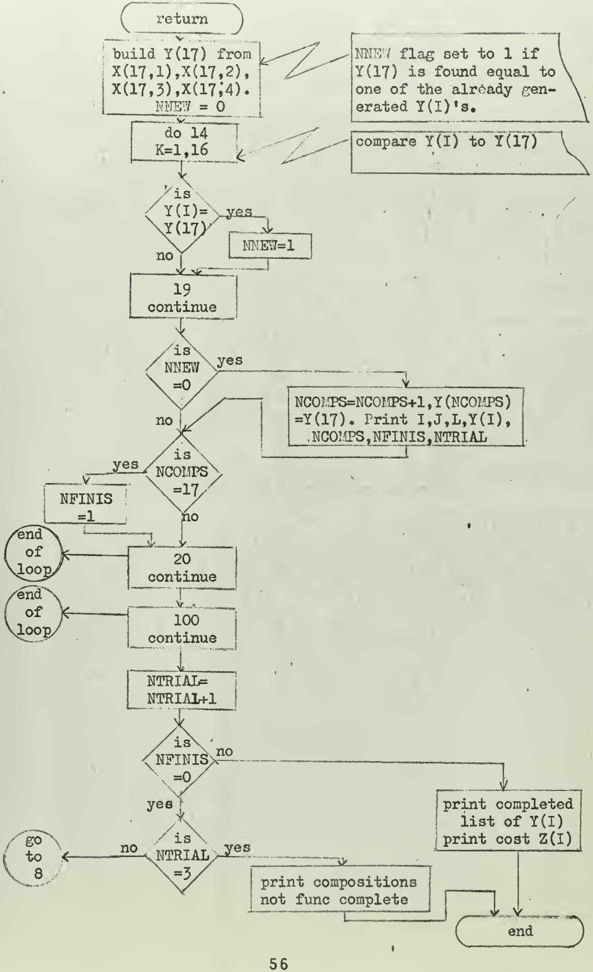
will operate on Y(I) and Y(J) with each of the given compositions to create new compositions, Y(17).

expedite do loop if finished.

Y(I) and Y(J) will be operated on by each given composition in the subroutine. NFUNCS is the # of times necessary to enter the subroutine to have all Y(I)s and Y(J)s operated on be each given composition.

Subroutine Derivl




return
build Y(17) from X(17,1),X(17,2), X(17,3),X(17,4). NNEW = 0
NNEW flag set to 1 if Y(17) is found equal to one of the already generated Y(I)'s.
do 14 K=1,16
compare Y(I) to Y(17)
is Y(I)= Y(17)
yes
NNEW=1
no
19 continue
is NNEW =0
yes
NCOMPS=NCOMPS+1,Y(NCOMPS) =Y(17). Print I,J,L,Y(I), NCOMPS,NFINIS,NTRIAL
no
is NCOMPS =17
yes
NFINIS =1
no
end of loop
20 continue
end of loop
100 continue
NTRIAL= NTRIAL+1
is NFINIS =0
no
print completed list of Y(I) print cost Z(I)
yes
is NTRIAL =3
no
go to 8
yes
print compositions not func complete
end

```
..JOB FRUCHTENICHT       RCX 126
      PROGRAM FC
C        THIS PROGRAM TESTS BINARY COMPOSITIONS FOR FUNCTIONAL
C     COMPLETENESS. IT WILL TAKE AS INPUTS ANY NUMBER OF GIVEN
C     COMPOSITIONS SUCH AS THE (AND) AND THE (NOT), DENOTED AS
C     (0001) AND (1010), AND OPERATE ON THESE INPUTS IN SUCH A
C     WAY THAT NEW COMPOSITIONS ARE GENERATED.  IF ALL SIXTEEN
C     POSSIBLE COMPOSITIONS ARE THUS GENERATED, THEN THE FIRST
C     AND MOST DIFFICULT TEST FOR FUNCTIONAL COMPLETENESS  FOR
C     THESE PARTICULAR INPUT COMPOSITIONS IS COMPLETED.
C
C     THE INPUT CARDS ARE AS FOLLOWS,
C     FIRST,  NCOMPS---NUMBER OF INPUT COMPOSITIONS.  THE ZERO
C             COMPOSITION SHOULD ALWAYS BE INCLUDED.  THERE ARE
C             TWO (NOT) COMPOSITIONS, NAMELY (1010) AND (1100).
C             NFUNCS-------NUMBER OF DIFFERENT OPERATIONS TO BE
C             PERFORMED.     FOR EXAMPLE, TESTING THE (AND) AND
C             THE (NOT), THEN NFUNCS EQUALS TWO.
C     SECOND, Y(I)---LIST OF ALL THE GIVEN INPUT COMPOSITIONS.
C     THIRD, Y(I)-----CONTINUATION OF SECOND IF MORE THAN NINE
C             COMPOSITIONS ARE GIVEN.   IF NOT, INSERT A BLANK.
C     FOURTH, X(I,J)-----ALSO A LIST OF GIVEN COMPOSITIONS BUT
C             NOW EACH INPUT COMPOSITION IS BROKEN UP INTO FOUR
C             INDIVIDUAL ELEMENTS.
C     FIFTH, Z(I)---AN APPROXIMATE RELATIVE COST OF GENERATING
C             EACH NEW COMPOSITION CAN BE OBTAINED BY INCLUDING
C             THE COST OF EACH INPUT COMPOSITION.
C     SIXTH, Z(I)------CONTINUATION OF FIFTH IF MORE THAN NINE
C             INPUTS. OTHERWISE INSERT BLANK.
C     SEVENTH, ---THIS IS THE FIRST OPERATION TO BE PERFORMED.
C             COST(I)  IS AN APPROXIMATION OF THE COST  FOR THE
C             OPERATION.   THE A1(I) IS THE RESULT DESIRED WHEN
C             OPERATING ON A ZERO AND A ZERO.  THE A2(I) IS THE
C             RESULT DESIRED WHEN OPERATING ON A ZERO AND A ONE
C             AND A3(I)  IS THE DESIRED RESULT FROM A ONE AND A
C             ONE.  FOR EXAMPLE, IF AN (AND) OPERATION IS TO BE
C             PERFORMED,  THEN A1(1) WOULD BE ZERO, A2(1) WOULD
C             BE ZERO,  AND A3(1) WOULD BE ONE.
C      EIGHTH, ---EACH ADDITIONAL CARD IS USED TO DENOTE A NEW
C             OPERATION.  THE NUMBER OF CARDS GREATER THAN SIX
C             MUST MATCH THE NUMBER OF NFUNCS.
C
C           THE OUTPUT WILL GIVE A NEW LISTING EACH TIME A NEW
C     COMPOSITION IS GENERATED.  THE NUMBERS GIVEN BY I AND J
C     WILL GIVE AN INDICATION TO THE TWO  COMPOSITIONS OF Y(I)
C     WHICH WERE USED TO GENERATE THE NEW ONE AND THE NUMBER L
C     WILL GIVE THE NUMBER OF THE OPERATION INVOLVED. THE LAST
C     PRINTOUT WILL INDICATE IF THE INPUT COMPOSITIONS  CAN BE
C     FUNCTIONALLY COMPLETE,  AND WILL ALSO GIVE AN INDICATION
C     OF THE COST INVOLVED.
      DIMENSION X(18,4),Y(18),Z(18)
      DIMENSION A1(16),A2(16),A3(16),A4(16)
      COMMON X,Y,Z,A1,A2,A3,COST
      READ 1,NCOMPS,NFUNCS
    1 FORMAT(2I3)
      READ 2,(Y(I),I=1,9)
    2 FORMAT(906)
      READ 3,(Y(I),I=10,16)
    3 FORMAT(706)
      READ 4,((X(I,J),J=1,4),I =1,8)
    4 FORMAT(3202)
      READ 5,(Z(I),I=1,9)
    5 FORMAT(9F6.2)
      READ 6,(Z(I),I=10,16)
    6 FORMAT(7F6.2)
      READ 7,(COST(I),A1(I),A2(I),A3(I),I=1,NFUNCS)
    7 FORMAT(F6.2,303)
      NFINIS =0
      NTRIAL =0
    8 DO 100 I=1,16
```

```
      DO 100 J=1,16
      IF(NFINIS) 100,9,100
    9 DO 23 L=1,NFUNCS
   10 CALL DERIV1(I,J,17,NCOMPS,L)
B  11 S=X(17,1)
B     T=X(17,2)
      LDA(S)              LDQ(T)
      QLS(45)             LLS(3)
      STA(R)              ENI(0)
B     S=X(17,3)
      LDA(R)              LDQ(S)
      QLS(45)             LLS(3)
      STA(R)              ENI(0)
B     S=X(17,4)
      LDA(R)              LDQ(S)
      QLS(45)             LLS(3)
      STA(T)              ENI(0)
B     Y(17)=T
      NNEW=0
      DO 14 K=1,16
      IF(NNEW) 12,12,14
B  12 IF  (-((Y(17)*Y(K))+((-Y(17))*(-Y(K)))))  14,13,14
   13 NNEW =1
   14 CONTINUE
      IF(NNEW) 15,15,23
   15 NCOMPS=NCOMPS +1
B     Y(NCOMPS)=Y(17)
      PRINT 16
   16 FORMAT(42H        I       J NCOMPS NFINIS NTRIAL      L)
      PRINT 17,I,J,NCOMPS,NFINIS,NTRIAL,L
   17 FORMAT(6I7/)
      PRINT 18
   18 FORMAT(X30H Y(01) Y(02) Y(03) Y(04) Y(05),
     124H Y(06) Y(07) Y(08) Y(09))
      PRINT 19,(Y(M),M=1,9)
   19 FORMAT(X,9(2X,O4)/)
      PRINT 20
   20 FORMAT(X24H Y(10) Y(11) Y(12) Y(13),
     124H Y(14) Y(15) Y(16) Y(17))
      PRINT 21,(Y(M),M=10,17)
   21 FORMAT(X,8(2X,O4)///)
      IF (NCOMPS-16) 23,22,22
   22 NFINIS=1
   23 CONTINUE
  100 CONTINUE
      NTRIAL=NTRIAL+1
      IF(NFINIS) 24,24,202
   24 IF (NTRIAL-3) 8,200,200
  200 PRINT 201
  201 FORMAT(42HCOMPOSITIONS ARE NOT FUNCTIONALLY COMPLETE)
      GO TO 210
  202 PRINT 203
  203 FORMAT(36HALL COMPOSITIONS HAVE BEEN GENERATED//)
      PRINT 204
  204 FORMAT(X30H Y(01) Y(02) Y(03) Y(04) Y(05),
     124H Y(06) Y(07) Y(08) Y(09))
      PRINT 205,(Y(M),M=1,9)
  205 FORMAT(X,9(2X,O4)/)
      PRINT 206,(Z(M),M=1,9)
  206 FORMAT(9(F5.2,1H$)/)
      PRINT 207
  207 FORMAT(X24H Y(10) Y(11) Y(12) Y(13),
     118H Y(14) Y(15) Y(16))
      PRINT 208,(Y(M),M=10,16)
  208 FORMAT(X,7(2X,O4)/)
      PRINT 209,(Z(M),M=10,16)
  209 FORMAT(7(F5.2,1H$))
  210 END
      SUBROUTINE DERIV1(I,J,NTEM,M,L)
      DIMENSION X(18,4),Y(18),Z(18)
```

```
        DIMENSION A1(16),A2(16),A3(16),A4(16)
        COMMON X,Y,Z,A1,A2,A3,COST
        DO 56 K=1,4
B    50 IF(X(I,K)) 51,51,54
B    51 IF(X(J,K)) 52,52,53
B    52 X(NTEM,K)=A1(L)
B       X(M+1,K)=A1(L)
        GO TO 56
B    53 X(NTEM,K)=A2(L)
B       X(M+1,K)=A2(L)
        GO TO 56
B    54 IF(X(J,K)) 53,53,55
B    55 X(NTEM,K) =A3(L)
B       X(M+1,K)=A3(L)
     56 CONTINUE
        Z(M+1)=COST(L)+Z(I)+Z(J)
        RETURN
        END
        END
```

NOT/AND COMPOSITIONS
INPUT DATA CARDS

```
 04 02
  1010   1100   0001   0000
  0000
 1 0 1  0 1 1  0 0 0  0 0 1  0 0 0 0
  2.00   2.00   2.00   0.00
  0.00
  2.00   0   0   1
  4.00   1   1   0
```

```
1..JOB FRIICHTENICHT      BOX 126
      I         J NCOMPS NFINIS NTRIAL       L
      1         1     5      0      0        2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  1010  1100  0001  0000  0101  0000  0000  0000  0000

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
  0000  0000  0000  0C00  0000  0000  0000  0101


      I         J NCOMPS NFINIS NTRIAL       L
      1         2     6      0      0        1

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  1010  1100  0001  0000  0101  1000  0000  0000  0000

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
  0000  0000  0000  0000  0000  0000  0000  1000


      I         J NCOMPS NFINIS NTRIAL       L
      1         2     7      0      0        2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  1010  1100  0001  0000  0101  1000  0111  0000  0000

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
  0000  0000  0000  0C00  0000  0000  0000  0111


      I         J NCOMPS NFINIS NTRIAL       L
      1         3     8      0      0        2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  1010  1100  0001  0000  0101  1000  0111  1111  0000

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
  0000  0000  0000  0000  0000  0000  0000  1111


      I         J NCOMPS NFINIS NTRIAL       L
      1         7     9      0      0        1

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  1010  1100  0001  0000  0101  1000  0111  1111  0010

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
  0000  0000  0000  0000  0000  0000  0000  0010


      I         J NCOMPS NFINIS NTRIAL       L
      1         7    10      0      0        2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  1010  1100  0001  0000  0101  1000  0111  1111  0010

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
  1101  0000  0000  0000  0000  0000  0000  1101


      I         J NCOMPS NFINIS NTRIAL       L
      2         2    11      0      0        2
```

```
Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
 1010  1100  0001  0000  0101  1000  0111  1111  0010

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
 1101  0011  0000  0000  0000  0000  0000  0011


     I       J NCOMPS NFINIS NTRIAL      L
     2       5     12      0      0      1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
 1010  1100  0001  0000  0101  1000  0111  1111  0010

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
 1101  0011  0100  0000  0000  0000  0000  0100


     I       J NCOMPS NFINIS NTRIAL      L
     2       5     13      0      0      2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
 1010  1100  0001  0000  0101  1000  0111  1111  0010

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
 1101  0011  0100  1011  0000  0000  0000  1011


     I       J NCOMPS NFINIS NTRIAL      L
     3       3     14      0      0      2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
 1010  1100  0001  0000  0101  1000  0111  1111  0010

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
 1101  0011  0100  1011  1110  0000  0000  1110


     I       J NCOMPS NFINIS NTRIAL      L
     7      14     15      0      0      1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
 1010  1100  0001  0000  0101  1000  0111  1111  0010

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
 1101  0011  0100  1011  1110  0110  0000  0110


     I       J NCOMPS NFINIS NTRIAL      L
     7      14     16      0      0      2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
 1010  1100  0001  0000  0101  1000  0111  1111  0010

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17)
 1101  0011  0100  1011  1110  0110  1001  1001


ALL COMPOSITIONS HAVE BEEN GENERATED

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
 1010  1100  0001  0000  0101  1000  0111  1111  0010
```

```
 2.00$ 2.00$ 2.00$  .00$ 8.00$ 6.00$ 8.00$ 8.00$12.00$

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16)
  1101  0011  0100  1011  1110  0110  1001

14.00$ 8.00$12.00$14.00$ 8.00$18.00$20.00$
 TIME,  0 MINUTES AND  23 SECONDS
1..END
```

APPENDIX II

Derivation of Three-valued One-place Functions

The following computer program tests to see if all 27 one-place functions can be generated from a set of given input compositions. The method used is similar to that used in Appendix 1, except with fewer inputs. In this case the only external inputs are the number of compositions being tested (NFUNCS) and the compositions themselves. For the Saturable Hall Logic Element discussed in the section on functional completeness, NFUNC = 2 and the first composition is (0,0,1,0,1,2,1,2,2). The second composition is (2,2,1,2,1,0,1,0,0). The more general non-commutative, compositional form is used in the program to make it more useful. For the above composition, the A's are assigned values as follows:

0 & 0 gives A1(1) = 0
0 & 1 gives A2(1) = 0
0 & 2 gives A3(1) = 1
1 & 0 gives A4(1) = 0
1 & 1 gives A5(1) = 1
1 & 2 gives A6(1) = 2
2 & 0 gives A7(1) = 1
2 & 1 gives A8(1) = 2
2 & 2 gives A9(1) = 2

Similarly, the second composition has A1(2) through A9(2) assigned the values (2,2,1,2,1,0,1,0,0), respectively. These compositions operate on the three internal constants (000), (111), and (222), and one other one-place function, (012). The first three could be generated d.c. power sources and since they are included, must be considered as part of the set of compositions being tested. The program could be modified quite easily to take out

these constants and have them generated by the program itself if so desired. The fourth internal input represents the one input to the devices in question. Looking at the printout, it is noted that, for the first new one-place function, Y(i) = Y(1) = (000) and Y(j) = Y(4) = (012). Now, with L = 1 indicating the first operator, (0,0,1,0,1,2,1,2,2), was used, the first new one-place function can be determined:

$$(000) \quad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 2 & 2 \end{array} \quad (012) = (001)$$

This function is recorded as Y(5). For the second function, i = 1, j = 4, and L = 2:

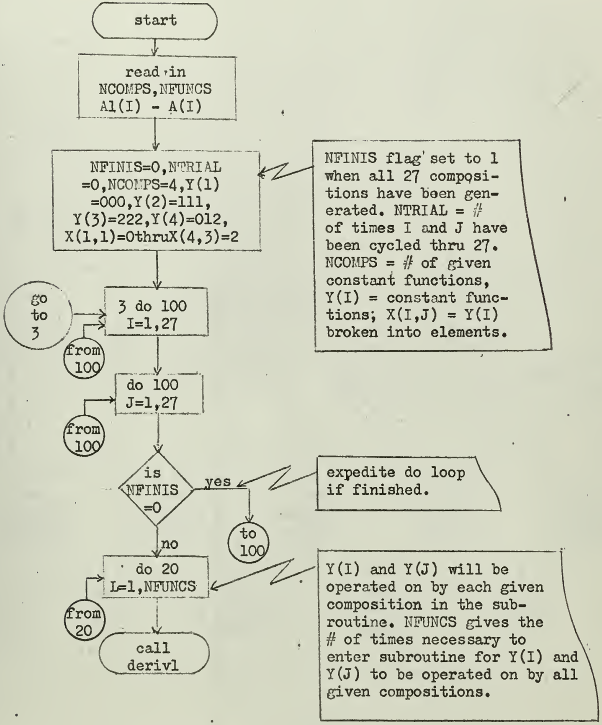
$$(000) \quad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 2 & 2 & 1 \\ 1 & 2 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{array} \quad (012) = (221)$$

In this manner all 27 one-place functions are formed.

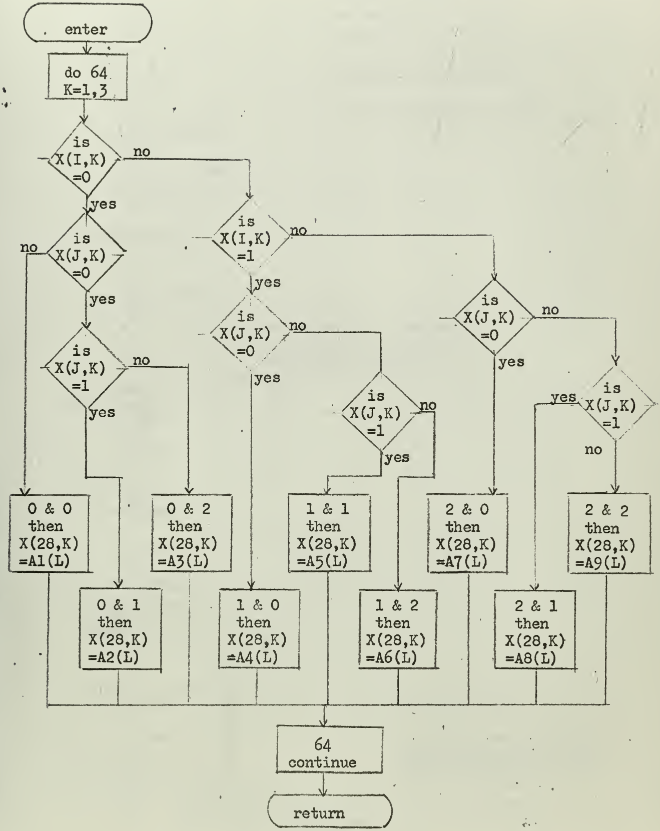
A print-out of a second set of compositions is included. This set is used in the section on an experimental adder circuit.

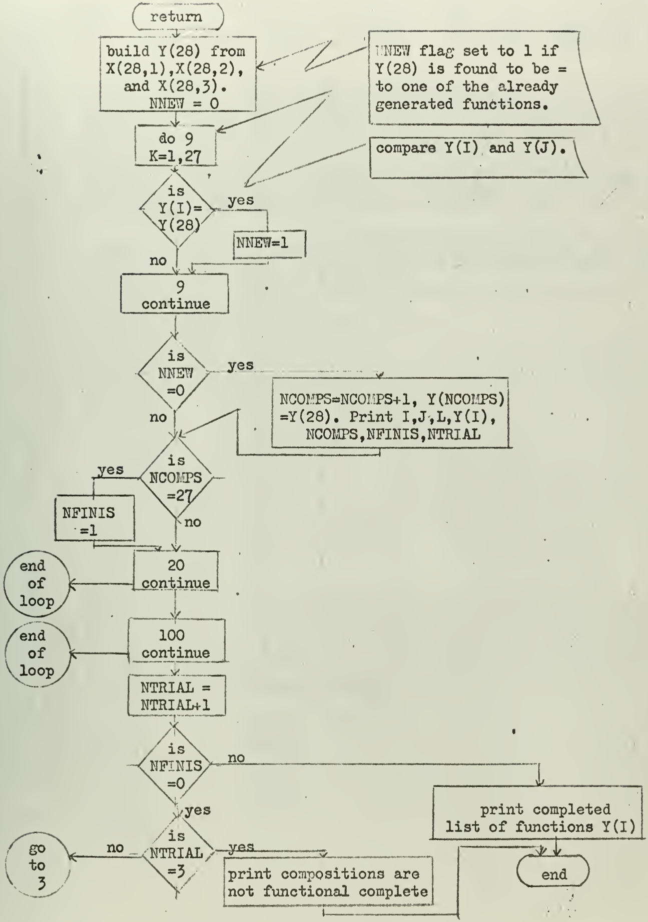
Appendix II, Fig. 1

Flow Chart for Deriving Three-valued One-place Functions


start
read in NCOMPS,NFUNCS A1(I) - A(I)
NFINIS=0,NTRIAL =0,NCOMPS=4,Y(1) =000,Y(2)=111, Y(3)=222,Y(4)=012, X(1,1)=0thruX(4,3)=2
NFINIS flag set to 1 when all 27 compositions have been generated. NTRIAL = # of times I and J have been cycled thru 27. NCOMPS = # of given constant functions, Y(I) = constant functions; X(I,J) = Y(I) broken into elements.
go to 3
3 do 100 I=1,27
from 100
do 100 J=1,27
from 100
is NFINIS =0
yes
to 100
expedite do loop if finished.
no
do 20 L=1,NFUNCS
from 20
call derivl
Y(I) and Y(J) will be operated on by each given composition in the subroutine. NFUNCS gives the # of times necessary to enter subroutine for Y(I) and Y(J) to be operated on by all given compositions.

SUBROUTINE DERIV1




return
build Y(28) from X(28,1),X(28,2), and X(28,3). NNEW = 0
NNEW flag set to 1 if Y(28) is found to be = to one of the already generated functions.
do 9 K=1,27
compare Y(I) and Y(J).
is Y(I)= Y(28)
yes
NNEW=1
no
9 continue
is NNEW =0
yes
NCOMPS=NCOMPS+1, Y(NCOMPS) =Y(28). Print I,J,L,Y(I), NCOMPS,NFINIS,NTRIAL
no
is NCOMPS =27
yes
NFINIS =1
no
20 continue
end of loop
100 continue
end of loop
NTRIAL = NTRIAL+1
is NFINIS =0
no
print completed list of functions Y(I)
yes
is NTRIAL =3
no
go to 3
yes
print compositions are not functional complete
end

```
..JOB FRIICHTENICHT      BCX 126
      PROGRAM FCT
C         THIS PROGRAM TESTS TO SEE IF ALL TWENTY-SEVEN THREE-
C       VALUED, ONE-PLACE FUNCTIONS CAN BE DERIVED BY OPERATING
C       ON CONSTANTS WITH DESIRED INPUT COMPOSITIONS.   THIS IS
C       THE FIRST AND THE MOST DIFFICULT  TEST  TO  CHECK OF THE
C       TWO CONDITIONS STATED BY SLUPEKI  THAT ARE  NECESSARY TO
C       BE SATISFIED FOR A GIVEN NUMBER  OF COMPOSITIONS  TO  BE
C       FUNCTIONALLY COMPLETE.
C
C       THE INPUT DATA CARDS ARE AS FOLLOWS,
C       FIRST, NFUNCS---THE NUMBER OF COMPOSITIONS TO BE TESTED.
C       SECOND, SIX NUMBERS REPRESENTING THE DESIRED OUTPUT FROM
C       A DEVICE, WITH THEM LISTED IN ORDER AS FOLLOWS, THE OUT-
C       PUT, A1,  THAT RESULTS FROM INPUTS OF A ZERO AND A ZERO.
C       OR A1=(0,0). ALSO A2=(0,1), A3=(0,2), A4=(1,1), A5=(1,2)
C       AND A6=(2,2).
C       THIRD,     ADDITIONAL CARDS ARE REQUIRED IF MORE THAN ONE
C       COMPOSITION IS BEING TESTED.   THE FORMAT IS THE SAME AS
C       THE SECOND CARD AND A NEW CARD  MUST BE  ADDED  FOR EACH
C       ADDITIONAL COMPOSITION.
C
C          THE OUTPUT WILL GIVE A LISTING OF ALL ONE-PLACE FUNC-
C       TIONS AS THEY ARE GENERATED. THE FUNCTIONS Y(I) AND Y(J)
C       AND THE OPERATION L CAN BE USED TO DETERMINE EXACTLY HOW
C       EACH NEW FUNCTION WAS GENERATED.
      DIMENSION A1(10),A2(10),A3(10),A4(10),A5(10),A6(10),
     1Y(28),XFUNC(29,3)
      COMMON A1,A2,A3,A4,A5,A6,Y,XFUNC
      READ 1,NFUNCS
    1 FORMAT(1I3)
      READ 2,(A1(I),A2(I),A3(I),A4(I),A5(I),A6(I),I=1,NFUNCS)
    2 FORMAT(603)
      NFINIS=0
      NTRIAL=0
      NCOMPS=4
B     Y(1)=000
B     Y(2)=111
B     Y(3)=222
B     Y(4)=012
B     XFUNC(1,1)=0
B     XFUNC(1,2)=0
B     XFUNC(1,3)=0
B     XFUNC(2,1)=1
B     XFUNC(2,2)=1
B     XFUNC(2,3)=1
B     XFUNC(3,1)=2
B     XFUNC(3,2)=2
B     XFUNC(3,3)=2
B     XFUNC(4,1)=0
B     XFUNC(4,2)=1
B     XFUNC(4,3)=2
    3 DO 100 I=1,27
      DO 100 J=1,27
      IF(NFINIS) 100,4,100
    4 DO 20 L=1,NFUNCS
    5 CALL DERIV1(I,J,NCOMPS,L,28)
B   6 S=XFUNC(28,1)
B     T=XFUNC(28,2)
      LDA(S)          LDQ(T)
      QLS(45)         LLS(3)
      STA(R)          ENI(0)
B     S=XFUNC(28,3)
      LDA(R)          LDQ(S)
      QLS(45)         LLS(3)
      STA(T)          ENI(0)
B     Y(28)=T
      NNEW=0
      DO 9 K=1,27
      IF(NNEW)7,7,9
```

```
B     7 IF(-((Y(28)*Y(K))+((-Y(28))*(-Y(K))))) 9,8,9
      8 NNEW=1
      9 CONTINUE
        IF(NNEW) 10,10,20
     10 NCOMPS=NCOMPS+1
B       Y(NCOMPS)=Y(28)
        PRINT 11
     11 FORMAT(//42H        I         J NCOMPS NFINIS NTRIAL        L)
        PRINT 12,I,J,NCOMPS,NFINIS,NTRIAL,L
     12 FORMAT(6I7/)
        PRINT 13
     13 FORMAT(X30H Y(01) Y(02) Y(03) Y(04) Y(05),
       124H Y(06) Y(07) Y(08) Y(09))
        PRINT 14,(Y(M),M=1,9)
     14 FORMAT(X,9(3X,O3)/)
        PRINT 15
     15 FORMAT(X30H Y(10) Y(11) Y(12) Y(13) Y(14),
       124H Y(15) Y(16) Y(17) Y(18))
        PRINT 16,(Y(M),M=10,18)
     16 FORMAT(X,9(3X,O3)/)
        PRINT 17
     17 FORMAT(X30H Y(19) Y(20) Y(21) Y(22) Y(23),
       124H Y(24) Y(25) Y(26) Y(27))
        PRINT 18,(Y(M),M=19,27)
     18 FORMAT(X,9(3X,O3)/)
        IF(NCOMPS-27) 20,19,19
     19 NFINIS=1
     20 CONTINUE
    100 CONTINUE
        NTRIAL=NTRIAL+1
        IF(NFINIS) 21,21,202
     21 IF(NTRIAL-3) 3,200,200
    200 PRINT 201
    201 FORMAT(42HCOMPOSITIONS ARE NOT FUNCTIONALLY COMPLETE)
        GO TO 210
    202 PRINT 203
    203 FORMAT(36HALL COMPOSITIONS HAVE BEEN GENERATED//)
        PRINT 204
    204 FORMAT(X30H Y(01) Y(02) Y(03) Y(04) Y(05),
       124H Y(06) Y(07) Y(08) Y(09))
        PRINT 205,(Y(M),M=1,9)
    205 FORMAT(X,9(3X,O3)/)
        PRINT 206
    206 FORMAT(X30H Y(10) Y(11) Y(12) Y(13) Y(14),
       124H Y(15) Y(16) Y(17) Y(18))
        PRINT 207,(Y(M),M=10,18)
    207 FORMAT(X,9(3X,O3)/)
        PRINT 208
    208 FORMAT(X30H Y(19) Y(20) Y(21) Y(22) Y(23),
       124H Y(24) Y(25) Y(26) Y(27))
        PRINT 209,(Y(M),M=19,27)
    209 FORMAT(X,9(3X,O3)/)
    210 END
        SUBROUTINE DERIV1(I,J,NCOMPS,L,NTEM)
        DIMENSION A1(10),A2(10),A3(10),A4(10),A5(10),A6(10),
       1Y(28),XFUNC(29,3)
        COMMON A1,A2,A3,A4,A5,A6,Y,XFUNC
        DO 64 K=1,3
B       XFUNC(29,1)=2
B    50 IF(XFUNC(I,K)) 51,51,56
B    51 IF(XFUNC(J,K)) 52,52,53
B    52 XFUNC(NTEM,K)=A1(L)
B       XFUNC(NCOMPS+1,K)=A1(L)
        GO TO 64
B    53 IF(XFUNC(J,K)*XFUNC(29,1)) 54,54,55
B    54 XFUNC(NTEM,K)=A2(L)
B       XFUNC(NCOMPS+1,K)=A2(L)
        GO TO 64
B    55 XFUNC(NTEM,K)=A3(L)
B       XFUNC(NCOMPS+1,K)=A3(L)
```

```
          GO TO 64
B    56   IF(XFUNC(I,K)*XFUNC(29,1)) 57,57,61
B    57   IF(XFUNC(J,K)) 54,54,58
B    58   IF(XFUNC(J,K)*XFUNC(29,1)) 59,59,60
B    59   XFUNC(NTEM,K)=A4(L)
B         XFUNC(NCOMPS+1,K)=A4(L)
          GO TO 64
B    60   XFUNC(NTEM,K)=A5(L)
B         XFUNC(NCOMPS+1,K)=A5(L)
          GO TO 64
B    61   IF(XFUNC(J,K)) 55,55,62
B    62   IF(XFUNC(J,K)*XFUNC(29,1)) 60,60,63
B    63   XFUNC(NTEM,K)=A6(L)
B         XFUNC(NCOMPS+1,K)=A6(L)
     64   CONTINUE
          RETURN
          END
          END
```

HALL EFFECT COMPOSITIONS

INPUT DATA CARDS

```
2
0  0  1  1  2  2
2  2  1  1  0  0
```

```
1..JOB FRIICHTLNICHT      BOX 126

      I        J NCOMPS NFINIS NTRIAL        L
      1        4     5      0      0         1

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   000   111   222   012   001   000   000   000   000

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   000   000   000   000   000   000   000   000   000

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   000   000   000   000   000   000   000   000   000


      I        J NCOMPS NFINIS NTRIAL        L
      1        4     6      0      0         2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   000   111   222   012   001   221   000   000   000

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   000   000   000   000   000   000   000   000   000

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   000   000   000   000   000   000   000   000   000


      I        J NCOMPS NFINIS NTRIAL        L
      1        6     7      0      0         1

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   000   111   222   012   001   221   110   000   000

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   000   000   000   000   000   000   000   000   000

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   000   000   000   000   000   000   000   000   000


      I        J NCOMPS NFINIS NTRIAL        L
      1        6     8      0      0         2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   000   111   222   012   001   221   110   112   000

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   000   000   000   000   000   000   000   000   000

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   000   000   000   000   000   000   000   000   000


      I        J NCOMPS NFINIS NTRIAL        L
      2        4     9      0      0         2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   000   111   222   012   001   221   110   112   210

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   000   000   000   000   000   000   000   000   000

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
```

```
  000   000   000   000   000   000   000   000   000

     I        J NCOMPS NFINIS NTRIAL       L
     3        4     10      0      0       1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   000   000   000   000   000   000   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000

     I        J NCOMPS NFINIS NTRIAL       L
     3        4     11      0      0       2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   000   000   000   000   000   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000

     I        J NCOMPS NFINIS NTRIAL       L
     3       11     12      0      0       1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   000   000   000   000   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000

     I        J NCOMPS NFINIS NTRIAL       L
     3       11     13      0      0       2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   000   000   000   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000

     I        J NCOMPS NFINIS NTRIAL       L
     4        5     14      0      0       1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   000   000   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
```

```
  000   000   000   000   000   000   000   000   000

     I        J NCOMPS NFINIS NTRIAL        L
     4        5     15      0      0        2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   000   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL        L
     4       10     16      0      0        1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL        L
     4       10     17      0      0        2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL        L
     4       15     18      0      0        1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   121

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL        L
     4       15     19      0      0        2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   121

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
```

```
  101   000   000   000   000   000   000   000   000

     I        J NCOMPS NFINIS NTRIAL       L
     7       13     20      0      0       1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   121

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  101   010   000   000   000   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL       L
     7       13     21      0      0       2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   121

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  101   010   212   000   000   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL       L
     7       16     22      0      0       1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   121

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  101   010   212   021   000   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL       L
     7       16     23      0      0       2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   121

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  101   010   212   021   201   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL       L
     7       18     24      0      0       1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   121

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
```

```
  101   010   212   021   201   120   000   000   000

    I        J NCOMPS NFINIS NTRIAL        L
    7       18     25      0      0        2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   121

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  101   010   212   021   201   120   102   000   000


    I        J NCOMPS NFINIS NTRIAL        L
    7       22     26      0      0        1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   121

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  101   010   212   021   201   120   102   020   000


    I        J NCOMPS NFINIS NTRIAL        L
    7       22     27      0      0        2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   121

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  101   010   212   021   201   120   102   020   202
ALL COMPOSITIONS HAVE BEEN GENERATED

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  000   111   222   012   001   221   110   112   210

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  122   100   211   011   002   220   022   200   121

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  101   010   212   021   201   120   102   020   202

TIME,  0 MINUTES AND  28 SECONDS
1..END
```

SET OF FOUR COMPOSITIONS
INPUT DATA CARDS

```
6

0  0  0  0  1  1  0  1  2
0  1  2  1  1  2  2  2  2
0  0  1  0  0  1  0  0  1
0  0  0  0  0  0  1  1  1
2  0  0  2  0  0  2  0  0
2  2  2  0  0  0  0  0  0
```

```
1   JOB FRIICHTENICHT      BOX 126

      I         J NCOMPS NFINIS NTRIAL       L
      1         1      5      0      0       3

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  012   111   222   000   001   000   000   000   000

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  000   000   000   000   000   000   000   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000


      I         J NCOMPS NFINIS NTRIAL       L
      1         1      6      0      0       5

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  012   111   222   000   001   200   000   000   000

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  000   000   000   000   000   000   000   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000


      I         J NCOMPS NFINIS NTRIAL       L
      1         2      7      0      0       1

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  012   111   222   000   001   200   011   000   000

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  000   000   000   000   000   000   000   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000


      I         J NCOMPS NFINIS NTRIAL       L
      1         2      8      0      0       2

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  012   111   222   000   001   200   011   112   000

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  000   000   000   000   000   000   000   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
  000   000   000   000   000   000   000   000   000


      I         J NCOMPS NFINIS NTRIAL       L
      1         5      9      0      0       5

Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
  012   111   222   000   001   200   011   112   220

Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
  000   000   000   000   000   000   000   000   000

Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
```

```
  000    000    000    000    000    000    000    000    000

       I        J NCOMPS NFINIS NTRIAL        L
       1        6     10      0      0        2

 Y(01)  Y(02)  Y(03)  Y(04)  Y(05)  Y(06)  Y(07)  Y(08)  Y(09)
   012    111    222    000    001    200    011    112    220

 Y(10)  Y(11)  Y(12)  Y(13)  Y(14)  Y(15)  Y(16)  Y(17)  Y(18)
   212    000    000    000    000    000    000    000    000

 Y(19)  Y(20)  Y(21)  Y(22)  Y(23)  Y(24)  Y(25)  Y(26)  Y(27)
   000    000    000    000    000    000    000    000    000

       I        J NCOMPS NFINIS NTRIAL        L
       1        6     11      0      0        3

 Y(01)  Y(02)  Y(03)  Y(04)  Y(05)  Y(06)  Y(07)  Y(08)  Y(09)
   012    111    222    000    001    200    011    112    220

 Y(10)  Y(11)  Y(12)  Y(13)  Y(14)  Y(15)  Y(16)  Y(17)  Y(18)
   212    100    000    000    000    000    000    000    000

 Y(19)  Y(20)  Y(21)  Y(22)  Y(23)  Y(24)  Y(25)  Y(26)  Y(27)
   000    000    000    000    000    000    000    000    000

       I        J NCOMPS NFINIS NTRIAL        L
       1        6     12      0      0        5

 Y(01)  Y(02)  Y(03)  Y(04)  Y(05)  Y(06)  Y(07)  Y(08)  Y(09)
   012    111    222    000    001    200    011    112    220

 Y(10)  Y(11)  Y(12)  Y(13)  Y(14)  Y(15)  Y(16)  Y(17)  Y(18)
   212    100    022    000    000    000    000    000    000

 Y(19)  Y(20)  Y(21)  Y(22)  Y(23)  Y(24)  Y(25)  Y(26)  Y(27)
   000    000    000    000    000    000    000    000    000

       I        J NCOMPS NFINIS NTRIAL        L
       1        9     13      0      0        1

 Y(01)  Y(02)  Y(03)  Y(04)  Y(05)  Y(06)  Y(07)  Y(08)  Y(09)
   012    111    222    000    001    200    011    112    220

 Y(10)  Y(11)  Y(12)  Y(13)  Y(14)  Y(15)  Y(16)  Y(17)  Y(18)
   212    100    022    010    000    000    000    000    000

 Y(19)  Y(20)  Y(21)  Y(22)  Y(23)  Y(24)  Y(25)  Y(26)  Y(27)
   000    000    000    000    000    000    000    000    000

       I        J NCOMPS NFINIS NTRIAL        L
       1        9     14      0      0        3

 Y(01)  Y(02)  Y(03)  Y(04)  Y(05)  Y(06)  Y(07)  Y(08)  Y(09)
   012    111    222    000    001    200    011    112    220

 Y(10)  Y(11)  Y(12)  Y(13)  Y(14)  Y(15)  Y(16)  Y(17)  Y(18)
   212    100    022    010    110    000    000    000    000

 Y(19)  Y(20)  Y(21)  Y(22)  Y(23)  Y(24)  Y(25)  Y(26)  Y(27)
```

```
   000   000   000   000   000   000   000   000   000

       I       J NCOMPS NFINIS NTRIAL       L
       1       9    15      0      0        5

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   000   000   000

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   000   000   000   000   000   000   000   000   000


       I       J NCOMPS NFINIS NTRIAL       L
       1      10    16      0      0        3

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   000   000

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   000   000   000   000   000   000   000   000   000


       I       J NCOMPS NFINIS NTRIAL       L
       1      13    17      0      0        5

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   000

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   000   000   000   000   000   000   000   000   000


       I       J NCOMPS NFINIS NTRIAL       L
       1      16    18      0      0        5

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   020

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   000   000   000   000   000   000   000   000   000


       I       J NCOMPS NFINIS NTRIAL       L
       2       6    19      0      0        2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   020

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
```

```
  211   000   000   000   000   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL        L
     2        9     20      0      0        2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   020

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   211   221   000   000   000   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL        L
     2       12     21      0      0        2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   020

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   211   221   122   000   000   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL        L
     2       18     22      0      0        2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   020

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   211   221   122   121   000   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL        L
     5        6     23      0      0        2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   020

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   211   221   122   121   201   000   000   000   000


     I        J NCOMPS NFINIS NTRIAL        L
     5       18     24      0      0        2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   020

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
```

```
  211   221   122   121   201   021   000   000   000

      I        J NCOMPS NFINIS NTRIAL        L
      6       13     25      0      0        2

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   020

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   211   221   122   121   201   021   210   000   000

      I        J NCOMPS NFINIS NTRIAL        L
      8       17     26      0      0        1

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   020

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   211   221   122   121   201   021   210   102   000

      I        J NCOMPS NFINIS NTRIAL        L
      9       21     27      0      0        1

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   020

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   211   221   122   121   201   021   210   102   120
ALL COMPOSITIONS HAVE BEEN GENERATED

 Y(01) Y(02) Y(03) Y(04) Y(05) Y(06) Y(07) Y(08) Y(09)
   012   111   222   000   001   200   011   112   220

 Y(10) Y(11) Y(12) Y(13) Y(14) Y(15) Y(16) Y(17) Y(18)
   212   100   022   010   110   002   101   202   020

 Y(19) Y(20) Y(21) Y(22) Y(23) Y(24) Y(25) Y(26) Y(27)
   211   221   122   121   201   021   210   102   120

 TIME,  0 MINUTES AND  37 SECONDS
1..END
```

thesF893
An experimental study of the uses of ter
3 2768 001 90051 7
DUDLEY KNOX LIBRARY